NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A MULTILEVEL SECURE CONSTRAINED INTRUSION DETECTION SYSTEM PROTOTYPE

by

Kah Kin Ang

December 2010

Thesis Co-Advisors: Cynthia E. Irvine
Thuy D. Nguyen

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 2010	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE A Multilevel Secure Constrained Intrusion Detection System Prototype	5. FUNDING NUMBERS
6. AUTHOR(S) Kah Kin Ang	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) DSTA	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number: N.A.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT

The Monterey Security Architecture (MYSEA) provides a distributed multilevel secure (MLS) environment consisting of a MLS local area network (LAN) and multiple single-level networks. The MYSEA server enforces a mandatory access control policy to ensure that users can only access data for which they are authorized. Intrusion detection systems (IDS) placed on a single-level network can store the alerts in the IDS databases at the same classification level as the network being monitored. As most databases do not support the enforcement of mandatory security policies, access to these databases is restricted to single-level access only. Thus, administrators are not presented with a coherent view of IDS alerts from all of the connected networks.

The objective of this thesis is to design a database proxy to allow administrators to view and analyze IDS information at multiple classification levels while enforcing the systems overall mandatory policy. Based on the derived concept of operations and the requirements, a design for the database proxy that mediates access to databases at different levels was conceived. A prototype database proxy was implemented along with modifications to a web-based analysis tool to allow the viewing and analysis of IDS information at multiple classification levels.

14. SUBJECT TERMS Intrusion Detection Systems (IDS), Information Assurance (IA), Monterey Security Architecture (MYSEA), Database Proxy	15. NUMBER OF PAGES 117
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

A MULTILEVEL SECURE CONSTRAINED INTRUSION DETECTION SYSTEM PROTOTYPE

Kah Kin Ang
Civilian, Naval Postgraduate School
B.Eng., Nanyang Technological University, 2004

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2010

Author: Kah Kin Ang

Approved by: Cynthia E. Irvine, PhD
Thesis Co-Advisor

Thuy D. Nguyen
Thesis Co-Advisor

Peter J. Denning, PhD
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Monterey Security Architecture (MYSEA) provides a distributed multilevel secure (MLS) environment consisting of a MLS local area network (LAN) and multiple single-level networks. The MYSEA server enforces a mandatory access control policy to ensure that users can only access data for which they are authorized. Intrusion detection systems (IDS) placed on a single-level network can store the alerts in the IDS databases at the same classification level as the network being monitored. As most databases do not support the enforcement of mandatory security policies, access to these databases is restricted to single-level access only. Thus, administrators are not presented with a coherent view of IDS alerts from all of the connected networks.

The objective of this thesis is to design a database proxy to allow administrators to view and analyze IDS information at multiple classification levels while enforcing the systems overall mandatory policy. Based on the derived concept of operations and the requirements, a design for the database proxy that mediates access to databases at different levels was conceived. A prototype database proxy was implemented along with modifications to a web-based analysis tool to allow the viewing and analysis of IDS information at multiple classification levels.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

BASE	Basic Security Analysis Engine
BLP	Bell-LaPadula
COTS	Commercial Off-the-Shelf
DAC	Discretionary Access Control
IDS	Intrusion Detection System
MAC	Mandatory Access Control
MLS	Multilevel Secure
MYSEA	Monterey Security Architecture
SSS	Secure Session Server
STOP	Secure Trusted Operating Program
TCM	Trusted Communications Module
TPE	Trusted Path Server

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to express my gratitude to both my advisors, Dr Cynthia Irvine and Thuy Nguyen, for their invaluable help and guidance throughout the entire course of this thesis. I would also like to thank Jean Khosalim for his help and technical assistance and Philip Hopfner for configuring and setting up the machines used for this thesis. Finally, I would like to thank my sponsors, Defence Science & Technology Agency (DSTA), for sending me to this course.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

Most databases are typically untrusted and do not support the enforcement of mandatory security policies. Once a connection has been established to a database, a user is allowed to read and write to it. For multilevel secure (MLS) systems that implement the Bell-LaPadula (BLP) security model [1], this behavior presents an access control problem. Specifically, the user is allowed to read data classified at and below his security level (no read-up). However, he is only allowed to write data at his security level and above it (no write-down). This presents a problem because once a user connects to a database classified below his security level, he has both read and write access. This is inconsistent with the BLP security model that is implemented in the MYSEA environment [2].

The Intrusion Detection System (IDS) in the MYSEA environment is deployed on multiple single-level networks. When Snort, which is IDS used in the MYSEA environment, raises an alert about a possible intrusion, a database client will write the alert into the database with the confidentiality level that is the same as that of the network. The Basic Security Analysis Tool (BASE), an open source web-based application, is then used to analyze the IDS data in the database [3]. As currently organized, both read and write access to the database that stores the intrusion detection records is required. This means that each instance of BASE is restricted to only read and analyze IDS data that is at its current level of confidentiality and not below it. To provide a more coherent view of the collected IDS data, the current IDS implementation in MYSEA must be modified to allow a user to read IDS data at and below the user's session level and to only write the consolidated IDS report at the current session level.

B. PURPOSE OF STUDY

The purpose of this thesis is to study and modify the existing MYSEA implementation to allow the BASE application read access to data that is classified below the level that the user is logged in. The following questions were examined in this thesis.

1. Can a trusted mechanism be designed that will permit operators to read intrusion detection information at multiple classification levels while enforcing the system's overall mandatory confidentiality policy?
2. The SQL database allows both read and write access when connected. Can the designed trusted mechanism restrict write access when a user at a higher security level connects to a database of a lower security level?

To provide a structured approach for this thesis, the following methodology is employed. The background of the existing implementation of the MYSEA environment is studied and components relating to this thesis are reviewed. The concept of operations and high-level requirements are developed to guide the design. This design is then implemented to demonstrate its proof of concept. The implementation is first conducted on a Linux-based system before porting it to the MYSEA server running on XTS400 system [4]. Testing methodologies are then employed in both developmental and acceptance testing phases to verify the correctness of the implementation. The results of this thesis project are intended allow the user logged in at a particular security level to read from an IDS database of a lower or equal security level while preventing write access to an IDS database of a lower security level.

C. ORGANIZATION OF THESIS

This thesis is organized as follows:

Chapter I presents the issues of accessing single-level databases in MLS systems, which provides the motivation and purpose of this thesis.

Chapter II provides some background information on the MYSEA environment, IDS, IDS analysis tools and SQL proxies, that forms the basis of this thesis.

Chapter III discusses the concept of operations that lead to the functional requirements and the design of the database proxy prototype and the modified IDS analysis tool, the Basic Analysis Security Engine (BASE). The selection process of the open source database proxy for this thesis is also presented.

Chapter IV covers the implementation of the database proxy in two phases. Phase 1 was first done on a Fedora 12 Linux system to gain familiarity with the database proxy and to avoid issues with the STOP 7 operating system. Phase 2 followed with the implementation of the database proxy prototype and the modified BASE application on the XTS400 system running the STOP 7 operating system.

Chapter V describes the test plan for the implementation presented in Chapter IV and the corresponding test results.

Chapter VI concludes the thesis with some possible future work on the IDS and database prototype and conclusion to the work done in this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

This chapter contains background information on the Monterey Security Architecture (MYSEA), the Secure Trusted Operating Program (STOP) Version 7, Intrusion Detection Systems (IDS) and the Basic Analysis and Security Engine (BASE).

A. MYSEA ENVIRONMENT OVERVIEW

This section covers an overview of the MYSEA environment as well as the STOP 7 operating system, on which MYSEA runs.

1. Overview

MYSEA provides a distributed multilevel secure (MLS) operating environment for the enforcement of mandatory security policies. It supports both mandatory access control (MAC) and discretionary access control (DAC). It consists of high-assurance components, such as the XTS400, which executes the STOP operating system and low-assurance commercial off-the-shelf (COTS) products. Figure 1 shows the layout of the MYSEA environment. The highly trustworthy Trusted Path Extensions (TPEs) and Trusted Communications Modules (TCMs) provide authentication and disambiguation for users and single level networks [2]. The TPE provides the trusted path between the users and the MYSEA servers, and provides a gateway between the COTS clients and the trusted server, while the TCM mediates the access of single level networks to the MYSEA servers. To support complex and adaptive operational environments, dynamic changes to the security services may be required. As such, Dynamic Security Services (DSS) mechanisms in MYSEA allow the modification of protection attributes of IPSec tunnels and access to application services [5].

MYSEA relies on the XTS-400 trusted computing platform and the STOP operating system to enforce the mandatory confidentiality and integrity access control policies and in most cases, the discretionary policies as well. The MYSEA

effort includes the construction of middleware services to support application services and some of these middleware elements must be trusted. Application services do not enforce MAC policy, but are constrained by the STOP enforcement. However, some services, such as the Wiki [6], enforce the application-specific DAC policy. The MYSEA approach allows for the minimization of the number and the extent of trusted components to meet high assurance requirements.

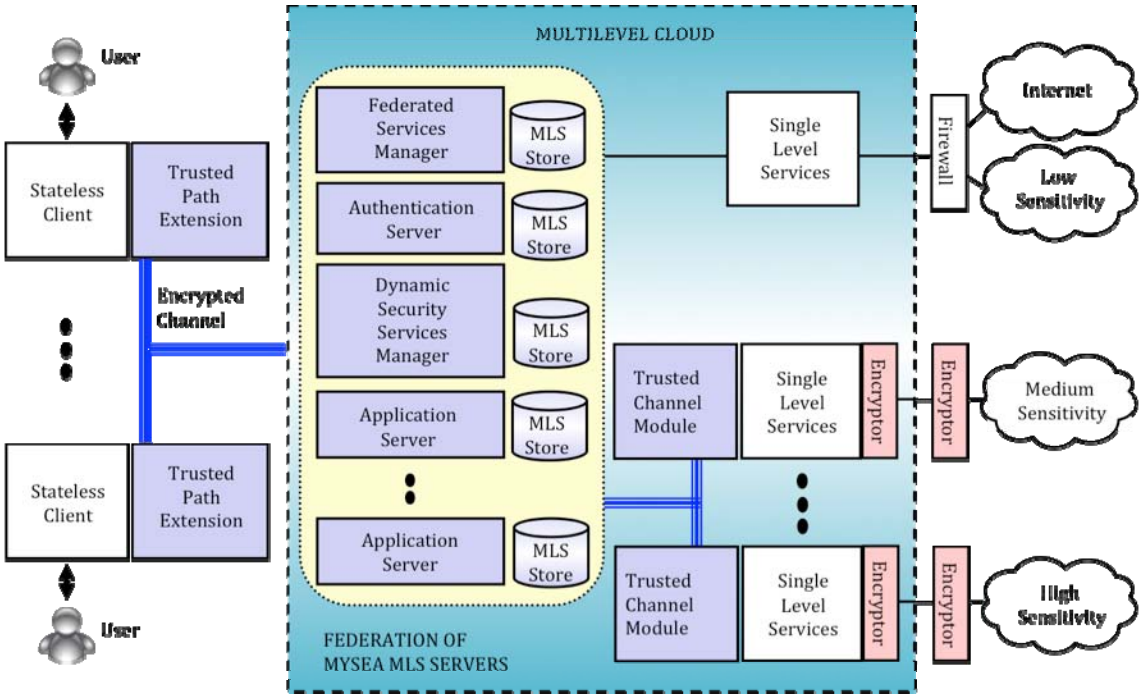

Figure 1. MYSEA environment

2. STOP 7 Operating System (OS)

The STOP 7 OS is the operating system used to power the XTS-400 trusted computing platform [4]. It enforces security policies based on the following access control models: role-based access control (RBAC), Bell-LaPadula (BLP) confidentiality model, Biba integrity model. These policies can be used separately or together to enforce an enterprise security policy for data protection. The version 7 of STOP OS provides an improvement on its performance and flexibility over its previous versions.

STOP 7 contains an auditing and system logging feature called “SLOG” (System LOGger). This provides safeguards to minimize the loss of audit data in the event of a system failure. STOP 7 also has additional capabilities for cryptographic support and a stateful packet filtering firewall at the kernel-level. These features allow the system to support security policy objectives and to be deployed for use in networked environments.

B. INTRUSION DETECTION SYSTEMS (IDS)

This section provides an overview of IDS functionality and its limitations. Two open source IDSes, Snort and Suricata are also discussed.

1. Overview of an IDS

Intrusion detection is the process of monitoring and analyzing system behavior or traffic anomalies in the system or network for possible threats of violation of computer security policies. An IDS is a device or software application that performs this functionality by comparing the network traffic to known attack signatures. If an attack is detected, the IDS will raise an alarm by generating an alert. The IDS used in the MYSEA environment is Snort [7].

Two detection techniques are used by Snort to evaluate the network packets. The first technique is a signature-based detection method. Signatures are preconfigured and predetermined attack patterns based on unique lines of code or strings. The IDS raises an alarm if the network packet contains data that matches the signatures. These signatures have to be constantly updated to mitigate new and emerging threats. The second method is a statistical anomaly-based detection technique. Normal acceptable network traffic behavior is evaluated and a baseline of normal activity is created. Any network traffic that performs outside of this baseline is considered a possible attack and generates an alert.

Both techniques have limitations and are susceptible to false positives and false negatives. A false positive occurs when the IDS raises an alarm when no

attack has taken place. In signature-based IDS, this may occur due to poorly designed signatures. In anomaly-based IDS, this may be due to the inability to provide a consistent performance baseline. A false negative occurs when an actual attack has taken place but the IDS does not generate any alert. For a signature-based IDS, this occurs when the signature for the attack is absent and it can be mitigated by constantly updating the attack signatures. For anomaly-based IDS, a false negative occurs if the intrusion activity is similar to normal acceptable network traffic behavior. Other than testing the attacks against the baseline profile and adjusting the baseline, it is difficult to mitigate the false negatives for anomaly-based IDS.

The alerts generated by the IDS sensors are usually stored in flat log files by default. These log files could potentially store millions of entries, which would overwhelm the analyst performing the analysis. To overcome this problem, the alerts are often stored in a central repository using relational databases such as PostgreSQL or Microsoft MSSQL. Using relational databases makes it easier to manipulate the data for analysis. In addition, the database products are usually packages with management tools for replication, backup and recovery.

The next two sections provide a discussion of two IDSes: Snort and Suricata. Snort currently is used in the MYSEA environment and has been available for about 12 years. Suricata is a relatively new open source project that aims to replace Snort as the IDS of choice.

2. Snort

Snort [7] is an open source network intrusion prevention and detection system. An intrusion detection and prevention system (IDPS) is similar to an IDS except that it is usually placed inline, as shown in Figure 2, and has the ability to block detected incoming attacks. An inline IPS device usually sits in between the network and the firewall. All packets entering and leaving the network will pass through the IPS. Hence, the IPS is able to block detected incoming threats. An IDS sits offline between the network and the firewall. The packets do not go

through the IDS device. Hence, it is able to monitor and send out alerts if threats are detected, but not block them. Snort was created by Martin Roesh as a packet sniffer tool in 1998 [8]. It is now open source software and supported by Roesh's company, Sourcefire.

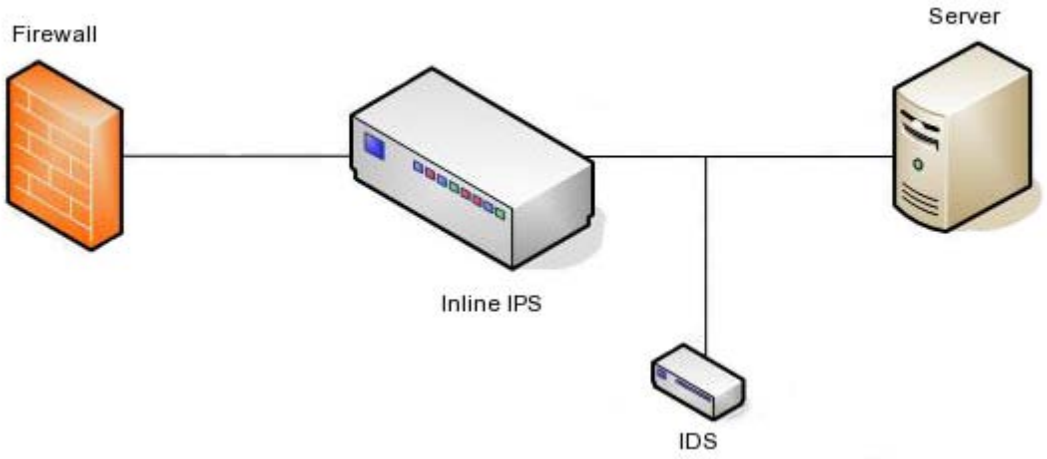

Figure 2. IPS and IDS placement

Snort's threat prevention components consist of a packet classifier, an IP defragmenter and a TCP reassembler, a portscan processor and a detection engine [9]. The packet classifier determines which packets are inspected. The IP defragmenter and TCP reassembler ensure that the packets are inspected in the correct order. The portscan processor watches for portscans and the detection engine performs protocol normalization, rule matching, and other detection functions to identify threats.

When Snort detects a threat, it will write an alert to an alert log by default. This alert logging can be configured to write to a syslog process or to a relational database such as PostgreSQL or Windows MSSQL Server. It can also be customized to write to unsupported databases or file formats.

3. Suricata

Suricata [10] is an open source IDS that provides support for the utilization of Snort's attack signatures. It was developed by the Open Information Security

Foundation (OISF) and a beta version was released on 31 December 2009. An advantage it has over Snort is that it supports multi-threading, which allows it to concurrently inspect multiple network packets. This improves the chances of detecting attack traffic. However, Suricata is still in the early phases of development and its performance is still slower than that of Snort [10]. Currently, it supports only a subset of the functionality offered by Snort. As such, Snort remains the choice of IDS for the MYSEA environment.

C. IDS ANALYSIS TOOLS

1. Overview of IDS Analysis Tools

IDSs tend to generate large volumes of log data for the alerts detected in the network. These log data allow analysts to understand the threats to the network and to provide a better defense of their systems. However, the huge volume of log data generated means that the analyst has to spend a large amount of time sifting through the individual records to identify the threats and attacks. IDS analysis tools help to alleviate this problem by automating some of the log analysis. The analysis tool used in the MYSEA environment is the Basic Analysis and Security Engine (BASE).

2. Basic Security Analysis Engine (BASE)

BASE [11] is a web-based interface written in PHP to analyze intrusions detected by the Snort IDS System. It is also able to search and process databases containing security events logged by different network monitoring tools. BASE has the ability to graphically display both Layer-3 (network) and Layer-4 (transport) data. Using parameters such as time, protocol, and classification, graphs and statistics can be generated and displayed.

The latest version of BASE (version 1.4.5) does not have any new features compared to the version (version 1.4.0) currently used in the MYSEA environment. However, there have been some security updates and bug fixes since the last version used in MYSEA. Currently, BASE is unable to display data

from multiple databases in a single web browser instance. Multiple browser instances have to be used to inspect and analyze information from multiple databases.

Development for BASE 2 has also started on 13 Aug 2010 [12]. It is based on the code from BASE 1.x and Analysis Console from Intrusion Databases project. This application will be able to analyze the alerts from a Snort or Suricata IDS system.

D. SQL PROXIES

The database used for MYSEA is PostgreSQL. The design presented in this work will use a SQL proxy. This section introduces SQL proxies and a particular proxy, PGPool-II.

1. Overview of SQL Proxies

SQL proxy services are processes that act as an intermediary between clients and SQL databases. SQL proxy services are able to accept requests from multiple clients, and forward them to the SQL servers. By doing so, all service requests to the database appear to come from a single source. There are various proxies, which perform different functions. For this project, PGPool-II will be used as the database proxy and will be modified to become MLS aware. The reasons for which PGPool-II was chosen are analyzed in the next chapter.

2. PGPool-II

PGPool-II is a proxy service that works between PostgreSQL servers and a PostgreSQL database client [13]. In addition to forwarding requests, it also provides the features shown in Table 1 [13].

Feature	Description
Connection Pooling	PGPool-II saves connections to the PostgreSQL servers and reuses them whenever a connection with the same properties is requested. This reduces the connection overhead.
Replication	PGPool-II performs replication by creating a real-time backup on two or more physical disks. This ensures that the service can still continue in the event of disk failure.
Load Balancing	PGPool-II utilizes the replication feature to reduce the load on each PostgreSQL server. This is done by distributing the SELECT queries among multiple servers, improving throughput.
Parallel Query	In this mode, data is distributed among multiple servers, so that a single query can be executed on all servers concurrently to reduce the overall execution time. A separate database is required to store the rules on how the data is distributed among the servers.

Table 1. PGPool-II features. From [13]

For this project, only the request-forwarding feature of the proxy is required. However, it is anticipated that the features described in Table 1 will be utilized to support federated database services in subsequent MYSEA releases.

3. SQL Relay and PL/Proxy

SQL Relay is a database proxy that provides persistent connection pooling, proxying and load balancing on Linux and Unix systems [14]. It provides services to speed up and enhance the scalability of database-driven web-based

applications, distributing accessing to replicated databases and migrating applications from one database to another.

PL/Proxy [15] is a database proxy that partitions databases and distributes the data across the partitions. Because the data is distributed, queries can be performed in parallel on several partitions to reduce the overall execution time.

Both proxies were evaluated together with PGPool-II for the purpose of this thesis. The selection criteria and outcome are described in Section E of Chapter III.

E. SUMMARY

Databases are not designed to support MLS systems, as the user is allowed to both read and write once a connection is made. As such, database proxies are required to mediate the connection between the user and the database. The next chapter will cover the concept of operation, requirements and design to illustrate the use of BASE and database proxies to mediate access to the IDS databases.

THIS PAGE INTENTIONALLY LEFT BLANK

III. REQUIREMENTS AND DESIGN

A. INTRODUCTION

This chapter will cover the concept of operations for a system that uses a database proxy to mediate the read and write operations to multiple IDS databases, each of which contains information at a different confidentiality level.

B. CONCEPT OF OPERATION

1. Database Proxy

The database proxy must operate in a way that does not violate the intended system policy. The intention of the database proxy is to allow a client to read IDS data stored at security levels that the current session level of the client dominates. It will also limit write access to only the security level that is equal to the current session level. This will align the access policy of the databases with the mandatory security policy enforced by STOP.

The BASE-native SQL client application will first send a SQL query to the database proxy. The database proxy will then check the client's current session level to determine if the requested query should be allowed. If the client's current session level is equal to the security level of the database, then the client is allowed to write to and read from the database. If the client's current session level is higher than the security level of the database, the client is only allowed to read from the database. The database proxy will allow this read-only functionality by filtering the SQL queries, and forwarding only the SELECT queries to the lower level databases.

2. BASE

BASE will be modified to allow the clients to decide which databases they want to access. However, only the allowable levels will be displayed to the users. For example, if the client is logged in at SIM_SECRET, then the option to access the SIM_SECRET or lower databases would be available. BASE will then

establish a connection to the databases through the respective database proxies. Figure 3 illustrates the concept of operations.

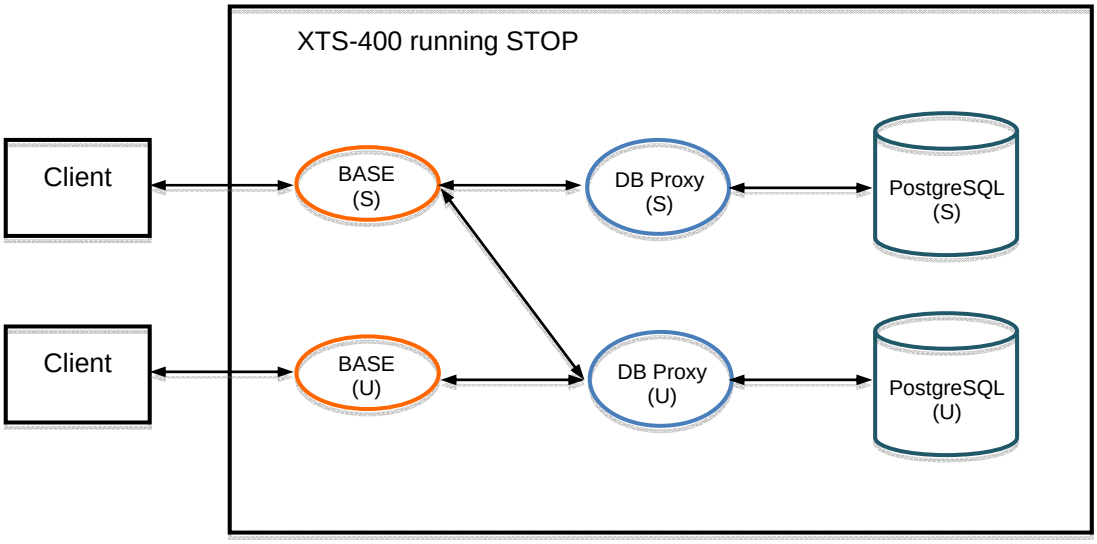

Figure 3. Concept of operations

BASE will also perform the consolidation of the reports if data from more than one database is requested. It will do this by first querying the databases through the respective database proxies, in the increasing order of security level. The results would be consolidated by appending them in the order of the query and displayed on the client. Each query must complete before the next query can be issued.

An advisory label will be shown at the top of the clients' display to inform the user of his current session level. Labels will also be inserted at the start and end of the data at each different security level. This will allow the user to clearly distinguish information at different sensitivity levels.

C. ARCHITECTURAL SECURITY ANALYSIS

In the original IDS design [3], the session level of the user was transparent to BASE. It was only able to connect to the database with the security label equal to the current session level. However, in the current design, BASE would be have

knowledge of the current session level and the databases it is allowed to access. Hence, some new threats to the system have to be addressed.

If the system was compromised, it might be possible for an adversary to modify BASE to access databases that have a higher security level than the current session level. Attempts to connect to those higher security level databases through the respective database proxies would be successful if the database proxy did not enforce the MAC policy. To mitigate this threat, the database proxy will check the current session against the security level of the database, and deny such attempts.

Another possible situation that could arise is that the adversary learns about the existence of the database proxies. The adversary could then attempt to connect BASE directly to the databases. However, the IP address and TCP ports of the databases are configured for single level accesses only. For single level accesses, only applications of that particular security level would be able to gain access. As such, the adversary would not be able to gain access to databases that are not equal to the current session level.

D. REQUIREMENTS

This section will cover the requirements to construct a working prototype based on the concepts of operations discussed in Section B.

1. Proxy Top-Level Requirements

The database proxy has the following requirements:

- The proxy shall only connect to the IDS database at the same security level at which the proxy runs. This connection shall be performed every time a query is requested from a SQL client application.
- The proxy shall accept IDS queries from a SQL client application.
- The proxy shall verify the security level of the client application prior to granting access to the IDS databases.

- The proxy shall allow the client application to query IDS data from the databases if the users' session level dominates the security level of the databases.
- The proxy shall deny the client application's access to the database if the current session level is lower than the database it is trying to access.
- If the current session level of the client application is higher than the security level of the database, the proxy shall filter the queries and only forward the queries starting with the word SELECT. A read query does not contain strings that will cause changes to the database. The filtering shall be done by parsing the first word in the query string to determine if it is associated with a read query. If no such word is found, it is considered a write and the query shall be dropped.
- There shall be one instance of the database proxy per security level in the system.
- The proxy shall keep a log of all requests.

2. BASE Top-Level Requirements

The BASE application has the following requirements:

- BASE shall display the security levels of the databases that it is allowed to query based on the current session level.
- If multiple databases are queried, BASE shall consolidate and display the results in a single web page.
- BASE shall query the allowed IDS database on behalf of an authenticated user.
- On the completion of a query, BASE shall close the connection request.
- BASE shall run at the authenticated users' session level.

There shall be at least one instance of the BASE application per client.

E. DATABASE PROXY SELECTION CRITERIA

To avoid building the database proxy from scratch, some open source options for the proxy were examined. The general requirements and selection criteria for the database proxy are discussed in this section.

1. General Requirements

The database proxy chosen must be able to mediate the connection and query requests between the client and the database server. As such, it must appear to the clients as the server and appear to the server as the client.

2. Selection Criteria

The selection criteria used to evaluate the database proxies are as follows:

- Compatible with PostgreSQL: PostgreSQL is the database used in the MYSEA environment. As such, the database proxy must be able to support PostgreSQL.
- Open source: Only open source database proxies will be considered for this thesis to avoid licensing costs. This is either a “Yes” or “No” criterion.
- Platform compatibility: The database proxy should be compatible with popular Linux-based operating systems such as Fedora and Red Hat. It should also be compiled, installed and run on the STOP 7 operating system without requiring major changes. The ranking for platform compatibility is given below.

a. Ranking Platform Compatibility

1. Supports a minimal set of common operating systems
2. Supports some common operating systems

3. Supports many common operating systems
4. Supports most common operating systems

- Minimality: The database proxy should contain the minimal functionality required to support the proxy requirements (see section D.1) without additional features that are not used by MYSEA. The ranking for proxy minimality is given below.

b. Ranking of Proxy Minimality

1. Contains the required functionality with many unnecessary features
2. Contains the required functionality with some unnecessary features
3. Contains the required functionality with no unnecessary features

- User support and documentation: Open source programs are usually not very well documented. That will cause difficulties when setting up and configuring the database proxy. The ranking for the documentation is given below.

c. Ranking Support and Documentation

1. Very little documentation, no user forums, tutorials or examples
2. Some documentation, has some user forums, no tutorials or examples
3. Good documentation, has user forums, some tutorials or examples
4. Good to excellent documentation, active user forums, several clear tutorials or examples

3. Selection Process

Three open source database proxies were identified for use in the MYSEA environment. The comparison and selection criteria are summarized in Table 2.

Database Proxy	PostgreSQL Compliant	Open Source	Platform Compatibility	Mini-mality	Support & Documentation
PGPool-II	✓	✓	4	2	4
SQL Relay	✓	✓	4	1	4
PL/Proxy	✓	✓	4	2	4

Table 2. Selection criteria for database proxy

4. Selection Outcome

All the database proxies in Table 2 meet the general requirements and are very close in the terms of their scores. However, while SQL Relay supports multiple databases including PostgreSQL, it is mainly intended to support the Oracle database. PGPool-II and PL/Proxy were both developed to support mainly the PostgreSQL database, which is used in MYSEA. The PGPool-II and PL/Proxy also have fewer features beyond those required for this thesis compared to SQL Relay. While PGPool-II has more unnecessary features than PL/Proxy for the purpose of this thesis, it is anticipated that these could be utilized to support federated database services in the future. Based on these considerations, PGPool-II is the choice for implementation as the database proxy in this project.

F. DESIGN

This section covers the high-level design of the multilevel database proxy using PGPool-II and the MLS-constrained BASE.

1. PGPool-II

The “IDS read-down” concept of operations introduces a database proxy service into the MYSEA environment. For this project, the database proxy is implemented using the open source PGPool-II database middleware program. The modified PGPool-II will ensure enforcement of the mandatory policy by checking the current session level against the security level of the database each time an access is requested. It will also filter out the write requests if the current session level is not equal to the security level of the database. Figure 4 shows the implementation design with PGPool-II as the database proxy.

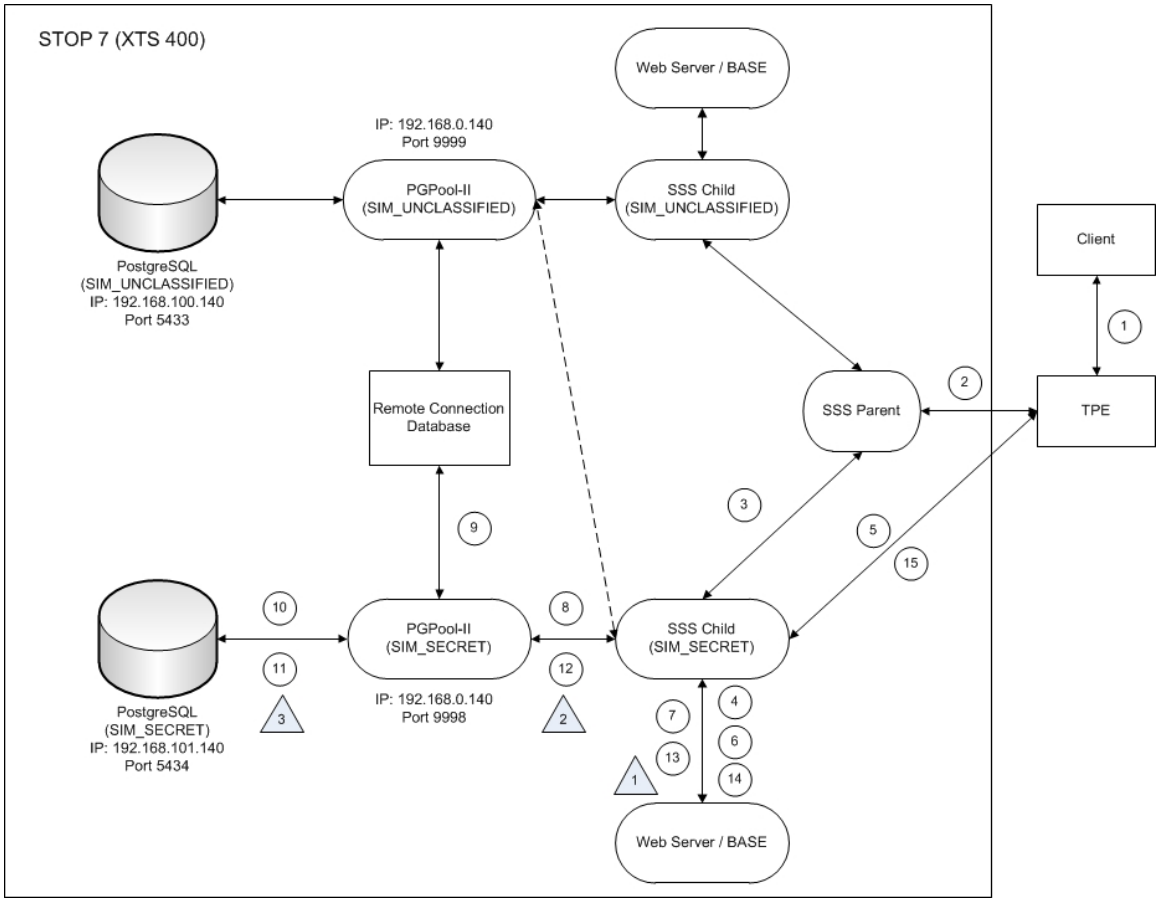

Figure 4. Database (PGPool-II) implementation design

Whenever the user logged in at the SECRET level requests for SIM_SECRET data from the IDS databases, the following occurs:

1. The client sends a request to the TPE.
2. The request is sent through the TPE to the Secure Session Server (SSS) Parent.
3. The SSS Parent first checks if the requesting client has an established session by searching for a matching entry in the User Database. If no entry is found, it then checks for a valid connection request by a remote application in the MYSEA Remote Connections Database. If the request is associated in either one of the two databases, an SSS Child is then spawned to handle the request.
4. The SSS Child then launches the Web Server, which runs at the current session level of the user or remote application.
5. The client invokes BASE through the TPE.
6. The SSS Child passes the request to the web server, which then executes BASE.
7. As BASE is untrusted, it is not given access to the MLS interface and must rely on the SSS Child that spawned it to make socket calls on its behalf. Hence, BASE requests for connection and queries to the PGPool-II (SIM_SECRET) proxy go through the SSS Child.
8. The SSS Child then establishes a connection to PGPool-II (SIM_SECRET) on the behalf of BASE and forwards all the connection requests and queries to PGPool-II (SIM_SECRET).
9. PGPool-II (SIM_SECRET) will then contact the Remote Connection database to perform a check to determine the session level associated with the request. Once the session level associated with the request is determined, PGPool-II (SIM_SECRET) will then compare the current session level against the security level of the requested IDS database. If the user's current session level dominates the security level of the IDS database, PGPool-II (SIM_SECRET) will allow the connection and

all read queries. If the users' session level equals the security level of the IDS database, it will allow all the write queries as well. Otherwise, the connection and all queries to the database will be denied. The queries are determined to be read queries if the strings within the queries start with the word "SELECT." Otherwise, they would be considered as write queries.

10. PGPool-II performs the connection and forwards the queries to the IDS database.
11. The IDS database returns the query result to PGPool-II (SIM_SECRET).
12. PGPool-II (SIM_SECRET) forwards the result back to the SSS.
13. The SSS Child, in turn, forwards the result to BASE for it to process.
14. BASE formats the results for display and returns the result to the SSS Child, after which, BASE will issue a request through the SSS to terminate the connection to the database.
15. The SSS Child will forward the formatted results back to the client through the TPE.

BASE will disconnect from the IDS database before connecting to another IDS database. This is done after step 14 and before step 15. The steps are displayed in triangles in Figure 4.

1. BASE will send a close connection request to the SSS.
2. The SSS will then forward it to PGPool-II (SIM_SECRET)
3. PGPool-II (SIM_SECRET) will forward it to the database to terminate the connection.

If access to a database with a security level lower than the current session level is requested (i.e., SIM_UNCLASSIFIED) from the same BASE session, steps 4 through 14 would be repeated. The difference is that the SSS child will

establish a connection to PGPool-II (SIM_UNCLASSIFIED) and PGPool-II (SIM_UNCLASSIFIED) will mediate all access to the database after that. The checks performed by PGPool-II (SIM_UNCLASSIFIED) in step 9 should result in allowing the connection to the database that PGPool-II (SIM_UNCLASSIFIED) manages and only read queries.

2. BASE

To support the changes made to the “IDS read-down” concept of operations, BASE has to be modified. The user should be able to view SQL databases at his current session level and choose the security level of the database he wants to view. If the user selects databases of different levels, the data will be displayed in ascending order of security levels. Figure 5 displays the new user interface design.

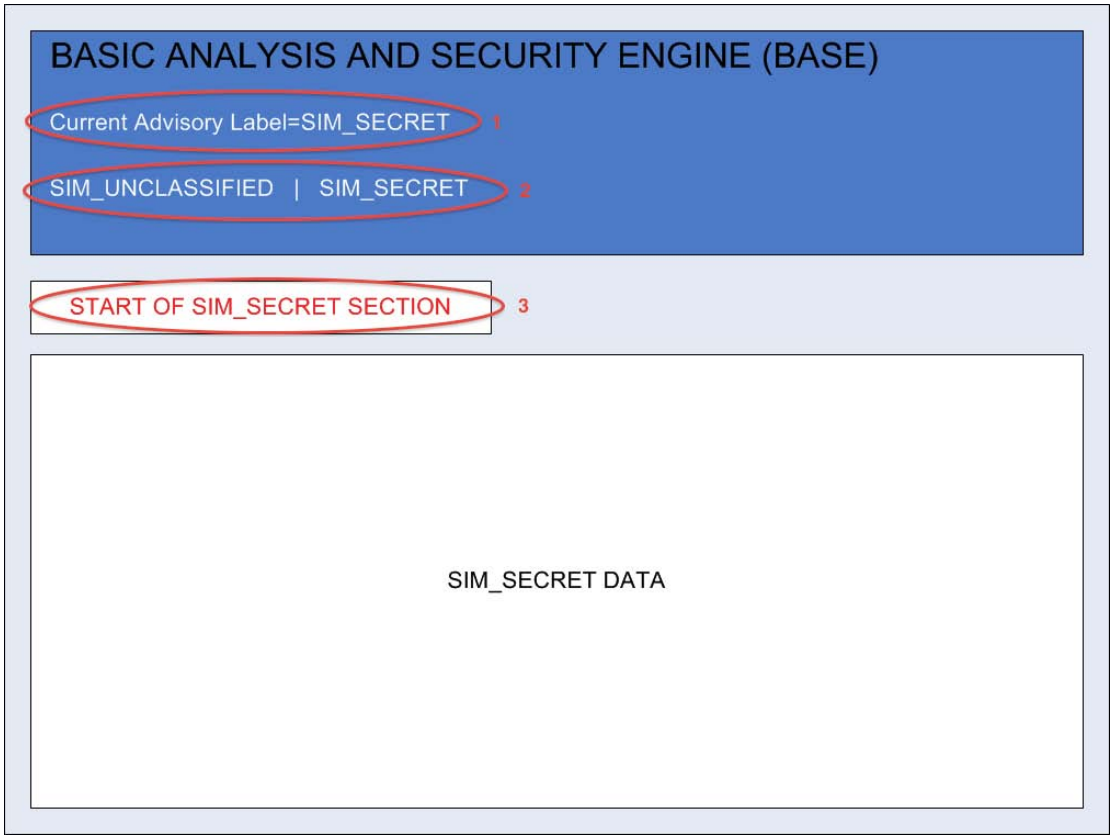

Figure 5. New user interface design

The circled region numbered 1 shows the current session level of the user. This information is obtained by executing a STOP system call to obtain the security level of the process that is serving BASE. Previously, there was no information regarding the session level of the user. The SQL database at the current session level is automatically queried and displayed after a successful BASE login. The circled region numbered 2 displays the security levels of the databases available to the user. Only security levels dominated by the current session level will be displayed. The circled region numbered 3 displays the security level of the data displayed below it. Figure 6 shows how the data is displayed if multiple security levels are selected. There are labels displayed at the beginning and end of each section to mark the start and end of information at a particular security level.

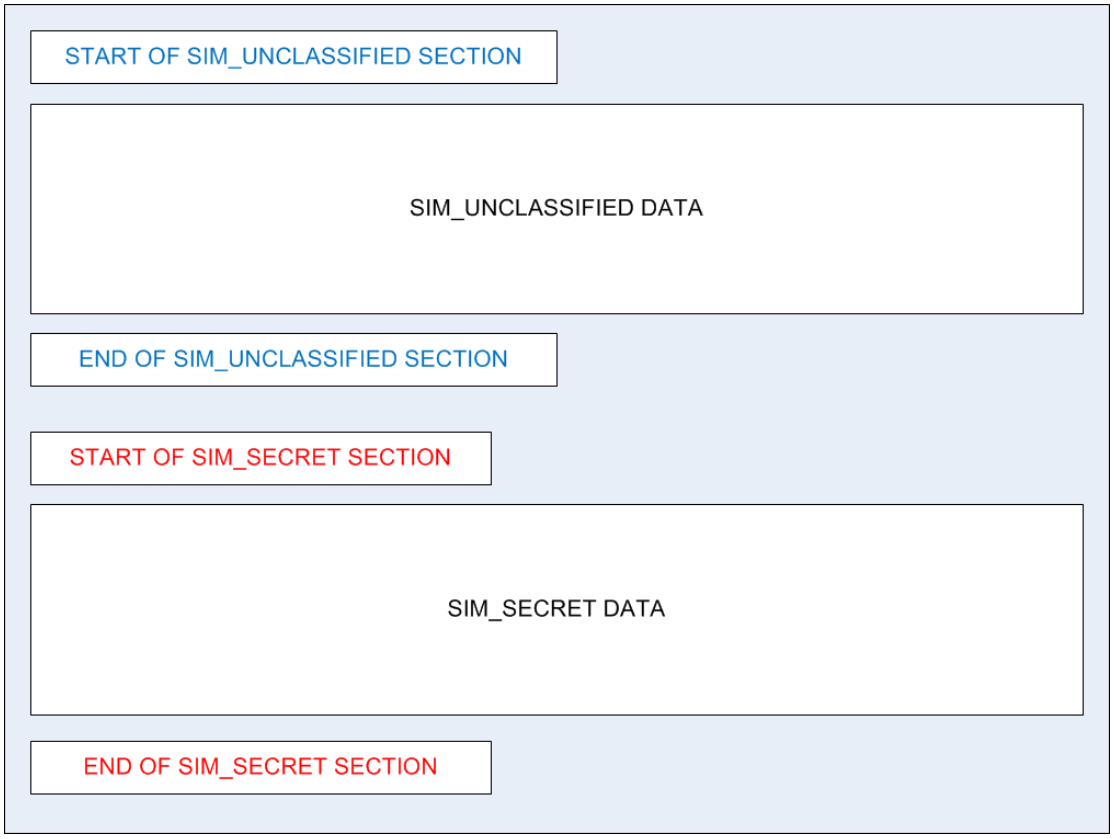

Figure 6. Design of BASE displaying data with two security levels

G. SUMMARY

The requirements analysis and design described in this chapter provide the supporting architecture for the implementation of the read-down and no write-down policy for databases. The no read-up policy is enforced by the STOP OS. A prototype for the project was implemented based on these requirements and the resulting design. The implementation is described in Chapter IV.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. IMPLEMENTATION

A. OVERVIEW

This chapter covers the implementation of the prototype as a proof-of-concept for the “IDS read-down” support for the MYSEA environment. The prototype is composed of a modified PGPool-II middleware program as the database proxy and a modified BASE application.

B. DEVELOPMENT ENVIRONMENT

The development of the prototype took place in two phases. The first phase was done on the Fedora 12 Linux operating system. The reason for performing the first phase on Fedora 12 was to gain familiarity with PGPool-II. This allowed possible problems associated with running PGPool-II on the STOP 7 OS to be avoided. For example, PGPool-II requires certain standard Linux functions (e.g., getuid()), which are not available in the STOP 7 OS environment.

In the first phase, PGPool-II was installed and configured to connect to two IDS databases with notational security levels SIM_UNCLASSIFIED and SIM_SECRET. A PostgreSQL client test program, *psql*, which is packaged with the PostgreSQL distribution, was used to test the connection with the two databases.

The second phase of the development was performed on the STOP 7 operating system. In this phase, the PGPool-II program was modified to be session-level-aware. The BASE application was also modified to support connections to multiple databases by using PGPool-II as the database proxy.

C. IMPLEMENTATION DETAILS

This section covers the details of PGPool-II implementation on the Fedora 12 and MYSEA operating environment. The implementation details for the BASE application on the MYSEA operating environment are also discussed.

1. PGPool-II on Fedora 12

PGPool-II requires that PostgreSQL libraries be installed. A copy of PostgreSQL 8.4.4 was obtained from www.postgres.org and installed. No configuration of the PostgreSQL 8.4.4 database is required, as only the library files are needed. Once the PostgreSQL libraries were installed, the installation of PGPool II can proceed. PGPool-II can be obtained from pgfoundry.org/projects/pgpool/. The detailed installation procedures for PGPool-II on the Fedora 12 operating system are covered in Appendix A.

The system was then set up as shown in Figure 7. Two instances of PGPool-II were run using different configurations. One instance was configured to connect to the SIM_SECRET database on the STOP 7 operating system while the other instance was configured to connect to the SIM_UNCLASSIFIED database.

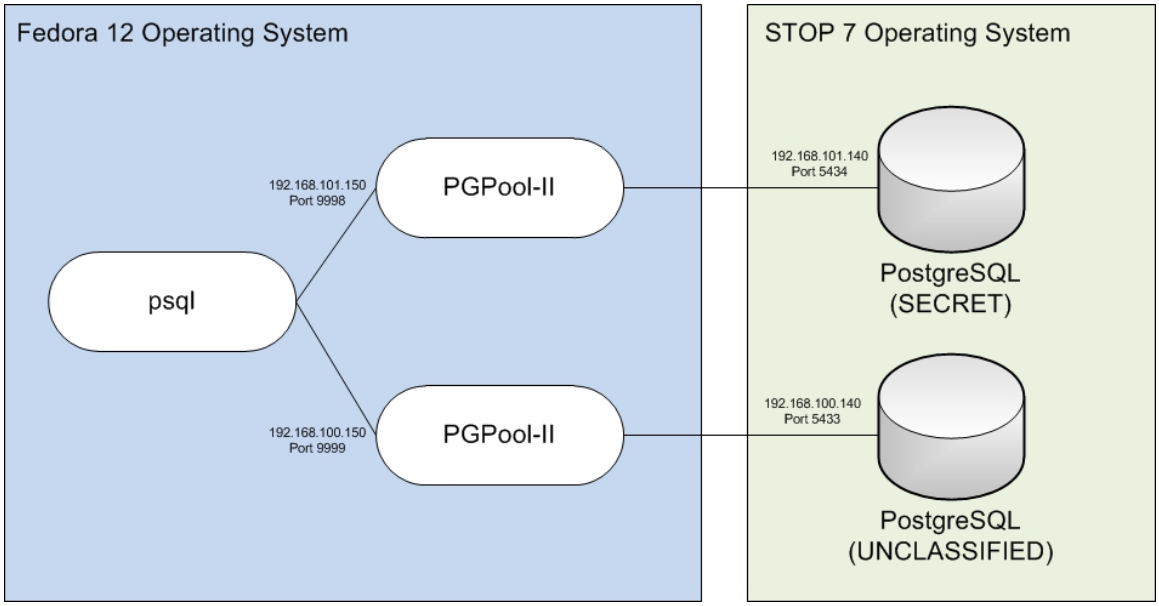

Figure 7. System setup with Fedora 12

The *psql* application, which was included in the PostgreSQL installation, was used to determine if the connection to the databases through the two

instances of PGPool-II worked. The *psql* application is a console-based front-end to PostgreSQL. It allows queries to be issued interactively to PostgreSQL, and obtains the results of those queries.

The *psql* application was connected to both instances of PGPool-II at the same IP address but different ports. In this configuration, queries made through PGPool at port 9998 would return results from the SECRET database, whereas queries made at port 9999 would return results from the UNCLASSIFIED database.

2. PGPool-II on MYSEA

PGPool-II was modified to be able to work as a database proxy as described in Chapter III. It has to be aware of the security level of the IDS database that it is serving, be able to know if the client's session level dominates the IDS database's level, and has to deny all write requests if the client's session level is not equal its session level. It also has to deny the connection if the IDS database's security level is higher than the client's session level. Modifications to the PGPool-II code for its start-up, connection and query requests processing were added.

Figure 8 shows the modification made to PGPool-II upon start up. The shaded boxes shows the added functionality for MYSEA. On start up, PGPool-II is invoked with a parameter specifying the security level of the IDS database. It then initializes access to the Remote Connection Module Database. If to the initialization sequence is successful, the PGPool-II proxy continues its normal execution. Otherwise, the PGPool-II proxy will terminate immediately.

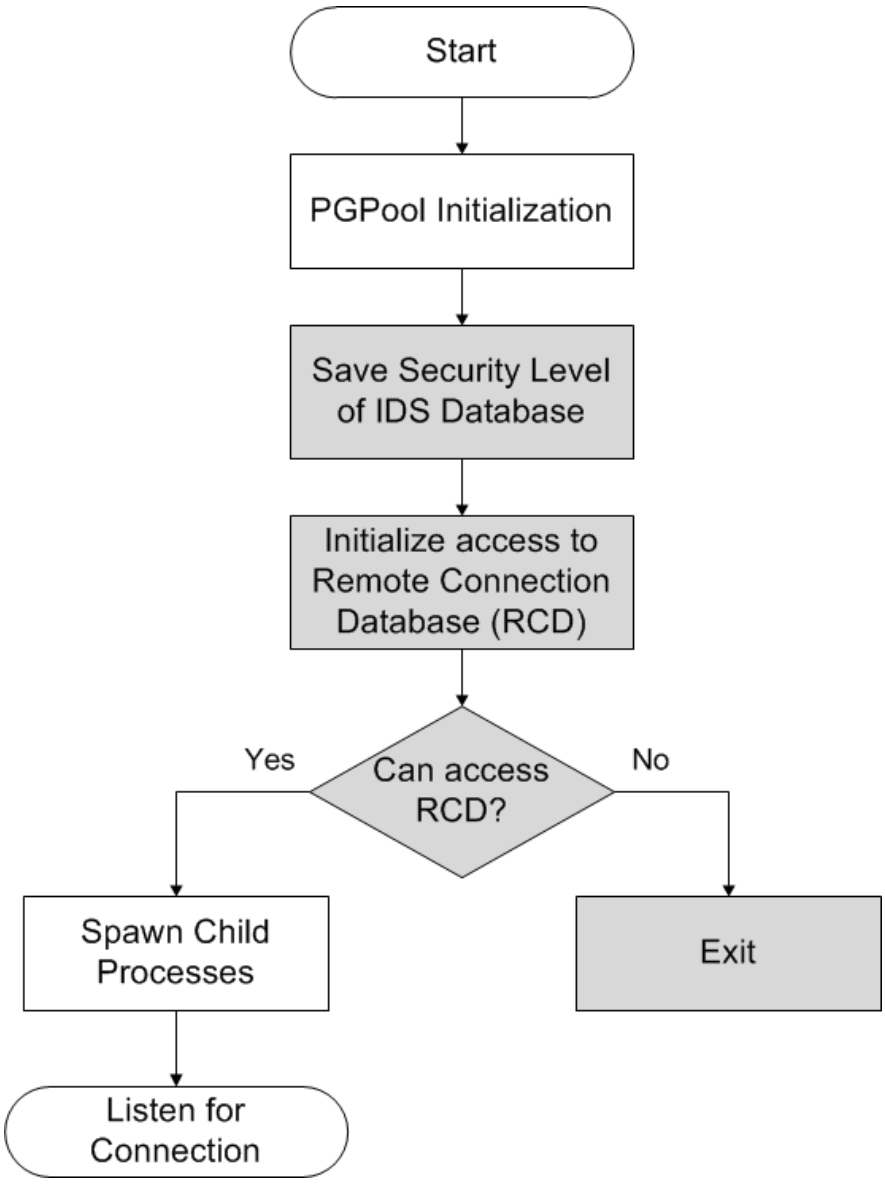

Figure 8. PGPool-II startup

Figure 9 shows the modified flow diagram to reflect the changes to PGPool-II during its connection processing. The shaded regions are the new additions to the PGPool-II application. PGPool-II will now check if the request is registered in the Remote Connection Database. If it is not, PGPool-II will deny the connection request to the database and resume listening for connections. If it is registered, PGPool-II will perform the connection to the database and start listening for queries.

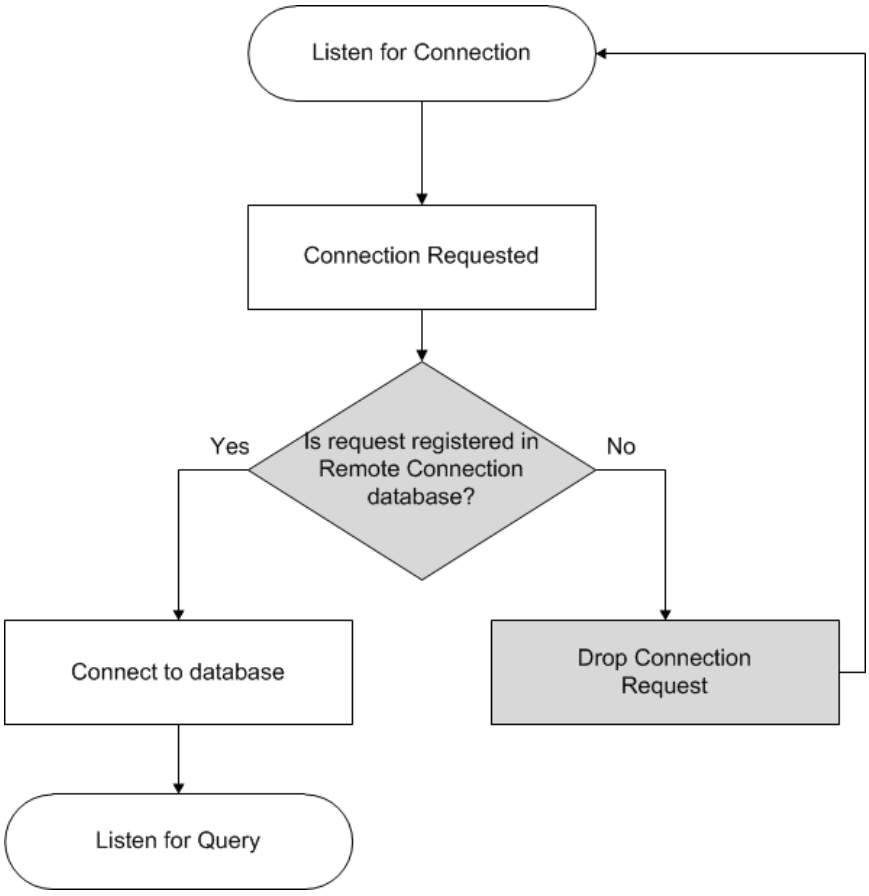

Figure 9. PGPool-II to database connection

The last modification to PGPool-II changes its handling of query requests. The shaded regions in Figure 10 show the changes to the logic made for the query request.

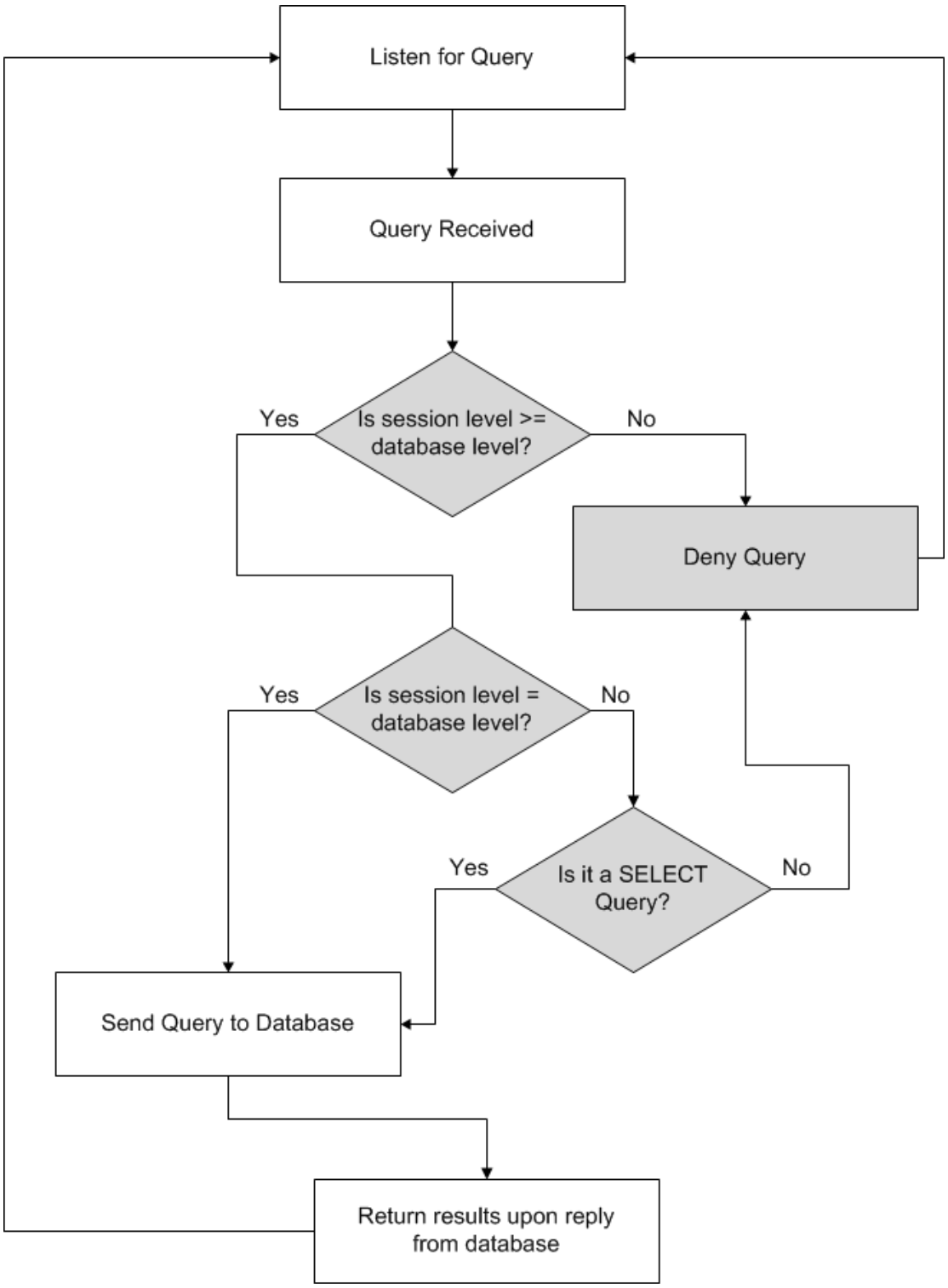

Figure 10. PGPool-II query request

On receiving a query request, PGPool-II will first check if the user's current session level dominates the security level of the database. If it does not, the query will be denied. If it does, PGPool-II will then check if the session level equals the security level of the database. If they are equal, the query will be forwarded to the database. Otherwise, PGPool-II checks if the request is a SELECT query. If it is a SELECT query, the query will be sent to the database. Otherwise, the query will be denied.

3. BASE on STOP 7

The BASE application was modified in order to support the "read-down" concept of operations needed for MYSEA. Headers have been added to the display to indicate the current session level and allowable display levels as shown in Figure 11.

Figure 11. BASE header display

The other major modifications to the BASE application are as follows:

- The configuration file includes an array of two entries of advisory label information: one for SIM_UNCLASSIFIED and the other for SIM_SECRET. Only two entries are used for the purpose of the demonstration; more can be added in the future. Each entry is a structure consisting of the security level, binary representation of the label, IP address and port number of the corresponding database, and color of the label for display. The details of each field for the advisory label information is given below.

    1. Security levels: This is the advisory label for the security level of the IDS database, e.g., SIM_UNCLASSIFIED.

2. Binary representation: This is a numerical value assigned to an advisory label to determine its dominance relation, e.g., SIM_UNCLASSIFIED and SIM_SECRET are assigned the value 0 and 2, respectively.
3. IP address: The IP address of the SQL database to connect to.
4. Port number: The port number of the SQL database to connect to.
5. Color: The color that the label would be displayed in at the start and end of that section.

- There is only one configuration file for BASE per system. The MAC and DAC permissions (in the form of MAC:DAC) for this file is set to *syslo:madmin.admin.0644*. This file cannot be modified by BASE during runtime.
- The execution of a STOP kernel call to extract the user's current session level.
- Advisory labels encapsulate the data displayed at each sensitivity level, as shown in Figure 12.
- If multiple IDS data with different security levels are selected for display, the data will be presented in separate sections, in the increasing order of security level, as shown in Figure 12.
- When a link in a particular section of a security level is selected, only the data related to that security level is presented in the new page in the same browser window. Advisory labels will also encapsulate requested data on the new page that BASE creates. All links within a section at a particular classification level point to information at that level.

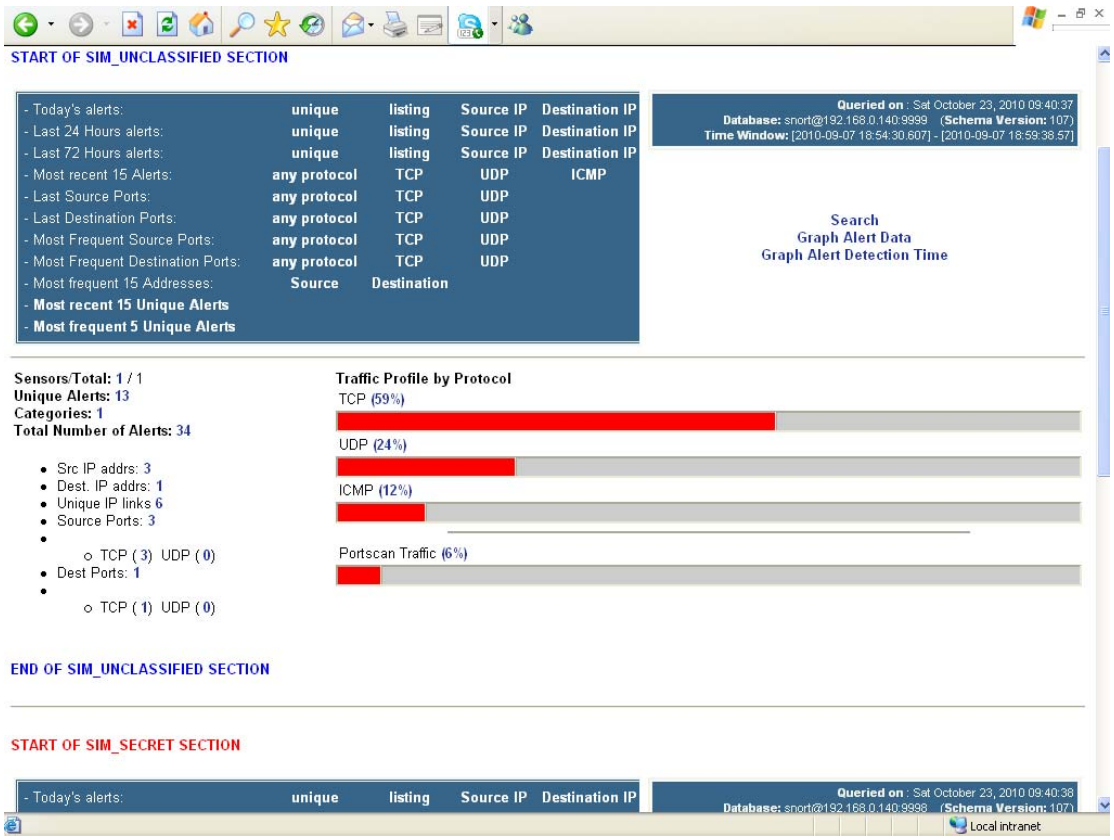

Figure 12. BASE displaying data with two security levels

Figure 13 displays the logic used to determine which advisory labels are to be added to the header of the BASE display. A STOP system call is first made to obtain the current user's session level (returned in ASCII string format), and saved for later use. The entries of advisory label information are then obtained from the configuration file and the binary representation of current session level is extracted by finding the matching security level string in those entries. This binary representation is then compared to all the mapping binary representation values contained in the entries. If the binary representation of the advisory label dominates the mapping binary representation value, the corresponding advisory label is added to the display. Otherwise, the advisory label is ignored. Once all the advisory labels have been compared, the header display is completed.

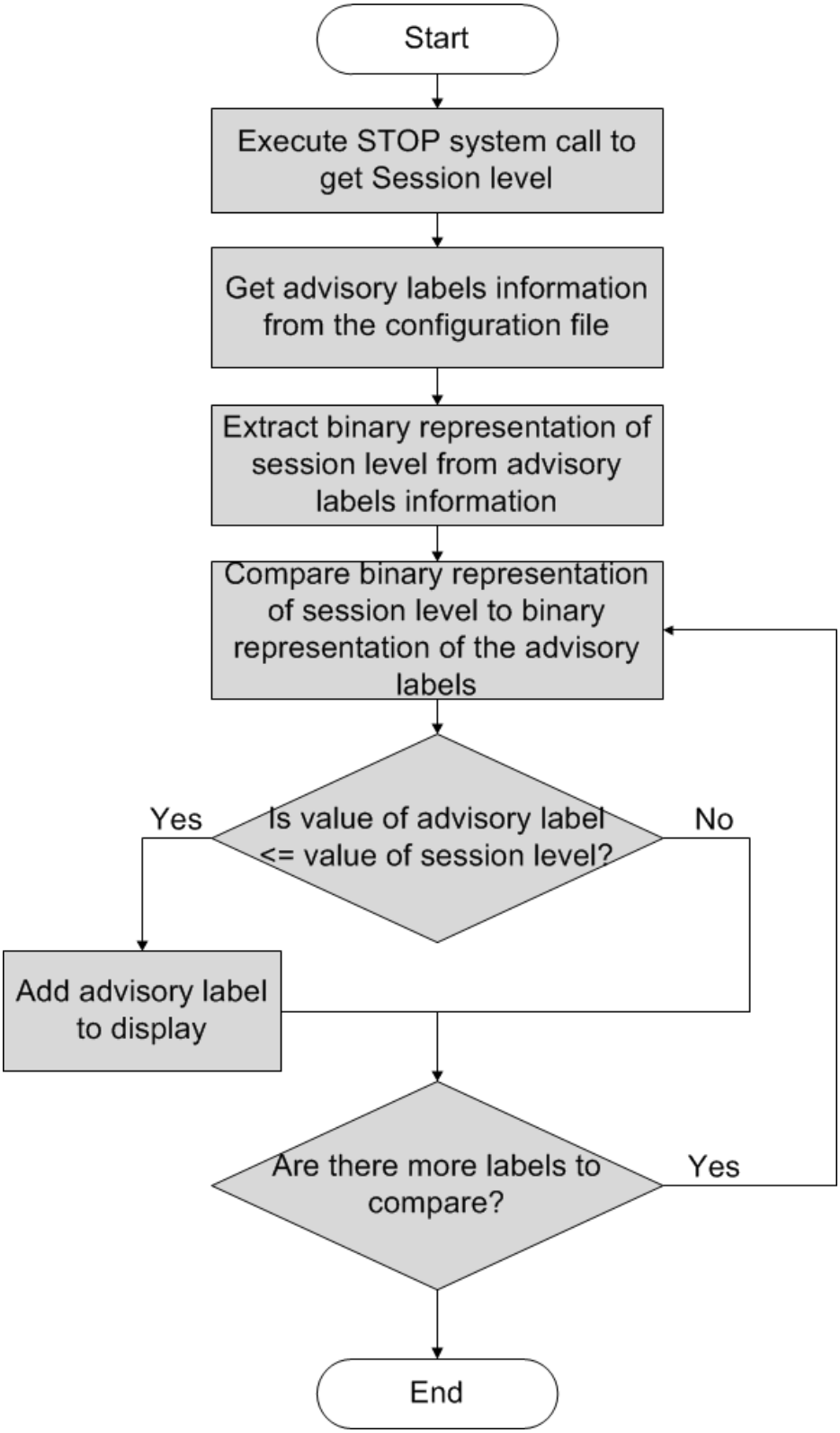

Figure 13. BASE header logic

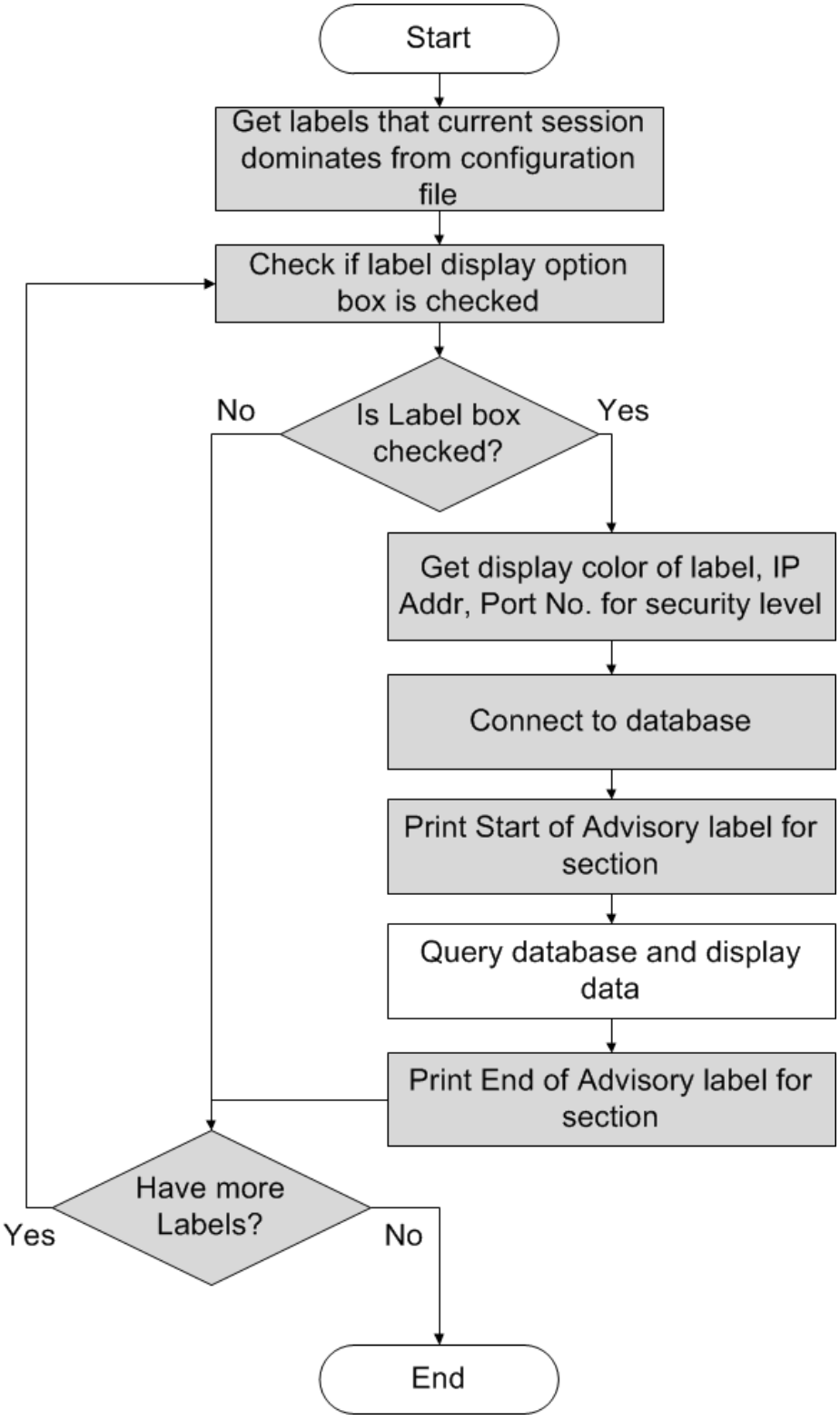

Figure 14. BASE data presentation logic

Figure 14 shows the logic used to query and present IDS data on the web browser. The shaded boxes show the changes to the display logic. The logic for the code is repeated based on the dominance relation of the labels. If the display checkbox for an advisory label that the current session level dominates is checked, it will read the connection parameters (IP address and port number), connect to the corresponding database, display the advisory label marking the start of the displayed data block, the data returned from the database and is completed with the second advisory label. The process is repeated for the each subsequent label dominated by the current session until there are no more labels.

D. PROBLEMS ENCOUNTERED

This section discusses some of the known problems with the current implementation and the problems encountered that were solved.

1. Issues With Current Implementation

In the current implementation, the BASE application will always perform a connection to the IDS database through the proxy before a query. As the connection to a lower level classification database is now allowed due to the database proxy, this leads to a potential covert channel using the connection requests. A malicious program could be used to cause the BASE application to connect to a database at timed intervals, resulting in a timing covert channel.

2. Solved Issues

Enabling automatic updates to the event cache require a write operation to the BASE-specific database. This would mean that if multiple IDS databases of varying security levels were accessed, this write operation would be performed on all the databases. As a write-down to the databases of lower security level is not allowed, this would result in the premature termination of the display due the error messages. A solution was to disable the automatic updates option in the configuration file. However, that results in the user having to manually perform

the update in order to view the new alerts. The updates to the databases of the lower security levels would also result in error messages as writing down is not permitted, causing a premature termination of the display.

A partial solution to this issue was to allow automatic updates to the event cache if the user's current session level is equal to security level of the database. Automatic updates to the event cache are not permitted if the security level of the database is lower than the user's current session level. To see if the event cache for the lower level database is updated, the user has to click on the "Cache & Status" link to display the cache status on a new web page. If the cache is not updated, the total number of events listed under "Alert Information Cache" will not tally with the cached events. Clicking on "Update Alert Cache" will cause an error as writing down is now permitted.

Figure 15. BASE event cache

In order to update the event cache, a modified concept of operations is required. The user has to log out of the current session level and log in at the level of the database with the event cache to be updated. The user then has to login to the BASE application, which will automatically update the event cache. The user can then log out of this session level and log in to the previous level, where he will be able to view the updated and consolidated IDS data.

E. SUMMARY

With the implementation of PGPool-II as the database proxy and modification of BASE, the no read-up and no write-down MAC policy is enabled. The reports generated from different security levels of IDS databases can also consolidated and displayed in a single page. The next chapter describes the developmental and acceptance tests performed to ensure that the modifications support the top-level requirements.

V. TESTING

This chapter describes the development and acceptance test plans designed to verify that the newly implemented database proxy and modified BASE functions properly. The results of the tests conducted are also presented.

A. OVERVIEW

Eleven systems were utilized during testing: the MYSEA server that hosts the database proxy, the modified BASE and the IDS databases, a Fedora Linux system as the Server Gateway, three Fedora Linux systems as the TPEs, three clients equipped with a web browser, two servers running the IDS and an attacker machine, which will simulate attack traffic for the IDS. Figure 16 shows the setup of the test environment. Different clients are used to demonstrate that the BASE display will be the same regardless of the platform used. Each individual client will connect to the MYSEA Server through its TPE and the gateway server.

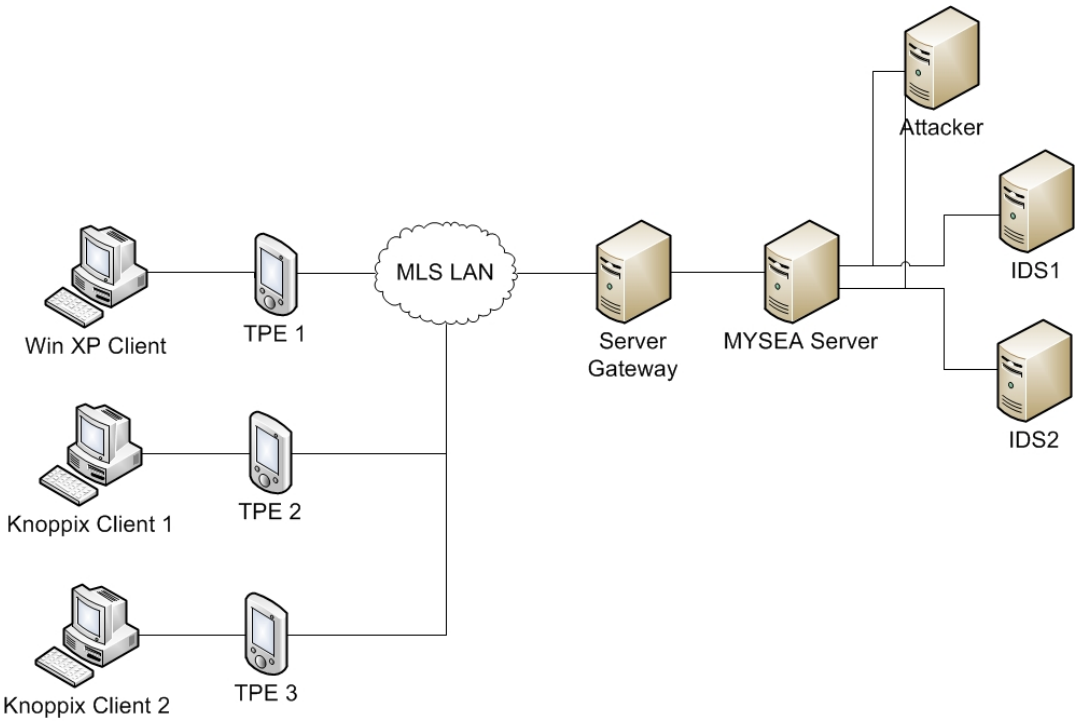

Figure 16. Test setup

The developmental and acceptance test plans and results in this chapter are only discussed at a general level; the detailed test procedures are documented in Appendix C.

B. DEVELOPMENTAL TESTING

The purpose of developmental testing is to test the functionality of each of the components of the database proxy and BASE that were implemented or modified as part of this thesis. Sample data in the IDS databases are used in this part of the testing.

1. PGPool-II Test Plan

The purpose of this test suite is to verify that the modified PGPool-II proxy correctly handles the connection to and query requests to the IDS databases. The web browser on the client machine is used to connect and query the IDS databases through BASE.

The tests are performed by trying to connect to the IDS databases at levels equal to, higher than and lower than the user's current session level. To perform the tests individually, the configuration file for BASE is modified such that it only has single level access. This is needed because using an unmodified configuration file will not allow tests that attempt to access databases at higher levels, such as the *connect up* test case (Test A7). The read and write queries to different security levels of the IDS databases are also tested. The test cases are described in Table 3.

Test ID	Test Type	Action	Expected Result
A1	Connect equal	Connect to IDS database with security level equal to that of the user's session level	Connection successful
A2	Read equal	Read request sent to the IDS database with a security level equal to that of the user's session level	Successful read query with results returned
A3	Write equal	Write request sent to the IDS database with a security level equal to that of the user's session level	Successful write query
A4	Connect down	Connect to the IDS database with a security level less than that of the user's session level	Connection successful
A5	Read down	Read request sent to the database with a security level less than that of the user's session level	Successful read query with results returned
A6	Write down	Write request sent to the IDS database with a security level less than that of the user's session level	Unsuccessful write query with error message returned.
A7	Connect up	Connect to the IDS database with a security level higher than that of the user's session level	Connection unsuccessful
A8	Incorrect parameters	Reload PGPool-II using incorrect IP address and port number to the IDS database.	Connection unsuccessful

Table 3. Test cases for PGPool-II

An additional exception test case was also performed to determine if unauthorized applications could to connect to the IDS databases through PGPool-II. This test is described in Table 4. Applications not registered in the Remote Connection Database are not allowed access to the IDS databases.

Test ID	Test Type	Action	Expected Result
B1	Unauthorized connection	Connect to IDS database using psql	Connection unsuccessful

Table 4. Exception test case for PGPool-II

2. BASE Test Plan

The purpose of this test suite is to verify that the modified BASE application correctly displays the user's current session level as an advisory label. It should also display all the security levels that are dominated by the user's current session level in the form of checkbox options. All the IDS data at each security level should be presented within sections marking the start and end of the IDS data of that security level. For this set of tests, BASE is set up as shown in Chapter III, Figure 4. The test cases are presented in Table 5.

Test ID	Test Type	Action	Expected Result
C1	Login Display at SIM_UNCLASSIFIED level	Display login page	Advisory label of SIM_UNCLASSIFIED is displayed. There is also only one option displayed for security level selection.
C2	Login at SIM_UNCLASSIFIED level	Login (which automatically queries the SIM_UNCLASSIFIED database)	Login Successfully. The main page is displayed with the advisory label of

			SIM_UNCLASSIFIED and the returned IDS data. There is also only one option displayed for security level selection. Advisory labels are placed at the start and end of the displayed IDS data section.
C3	Page Navigation for SIM_UNCLASSIFIED	Click on a link	A new page is displayed in the same window, showing the query results of the SIM_UNCLASSIFIED database. Advisory labels are placed at start and end of the SIM_UNCLASSIFIED IDS data.
C4	Login Display at SIM_SECRET level	Display login page	Advisory label of SIM_SECRET is displayed. There are also only two options (SIM_UNCLASSIFIED and SIM_SECRET) displayed for security level selection.
C5	Login at SIM_SECRET	Login (which automatically queries the SIM_SECRET database)	Login Successfully. Main page is displayed. Advisory Label of SIM_SECRET is displayed with the returned IDS data. Advisory labels are placed at the start and end of the displayed SIM_SECRET IDS data.
C6	Data consolidation	Ensure that both the SIM_UNCLASSIFIED and SIM_SECRET	Both checkboxes remain checked. Advisory labels for

		checkboxes are checked and click on the “Go” button.	SIM_UNCLASSIFIED and SIM_SECRET data are placed at the start and end of each section.
C7	Lower level data only	Ensure that only the SIM_UNCLASSIFIED checkbox is checked and click on the “Go” button.	The SIM_UNCLASSIFIED checkbox remains checked. Only the SIM_UNCLASSIFIED data is displayed and advisory labels for SIM_UNCLASSIFIED data are placed at the start and end of the IDS data section.
C8	Page Navigation - SIM_SECRET	Click on a link in the SIM_SECRET section of the data.	The resulting page will be displayed in the same window, showing only the SIM_SECRET data. The corresponding advisory labels are placed at the start and end of the IDS data section.
C9	Page Navigation (SIM_SECRET)	Click on a link in the SIM_UNCLASSIFIED section of the data.	The resulting page will be displayed in the same window, showing only the SIM_UNCLASSIFIED data. The corresponding advisory labels are placed at the start and end of the IDS data section.

Table 5. BASE test cases

For the second set of tests, BASE is configured to connect directly to the IDS databases without using the database proxy as shown in Figure 17. The test

cases presented in Table 6 are used to verify that read-down is not allowed if BASE is misconfigured to connect directly to the IDS databases or with incorrect parameters to the database proxy. The SSS Child performs the enforcement to prevent BASE from reading down.

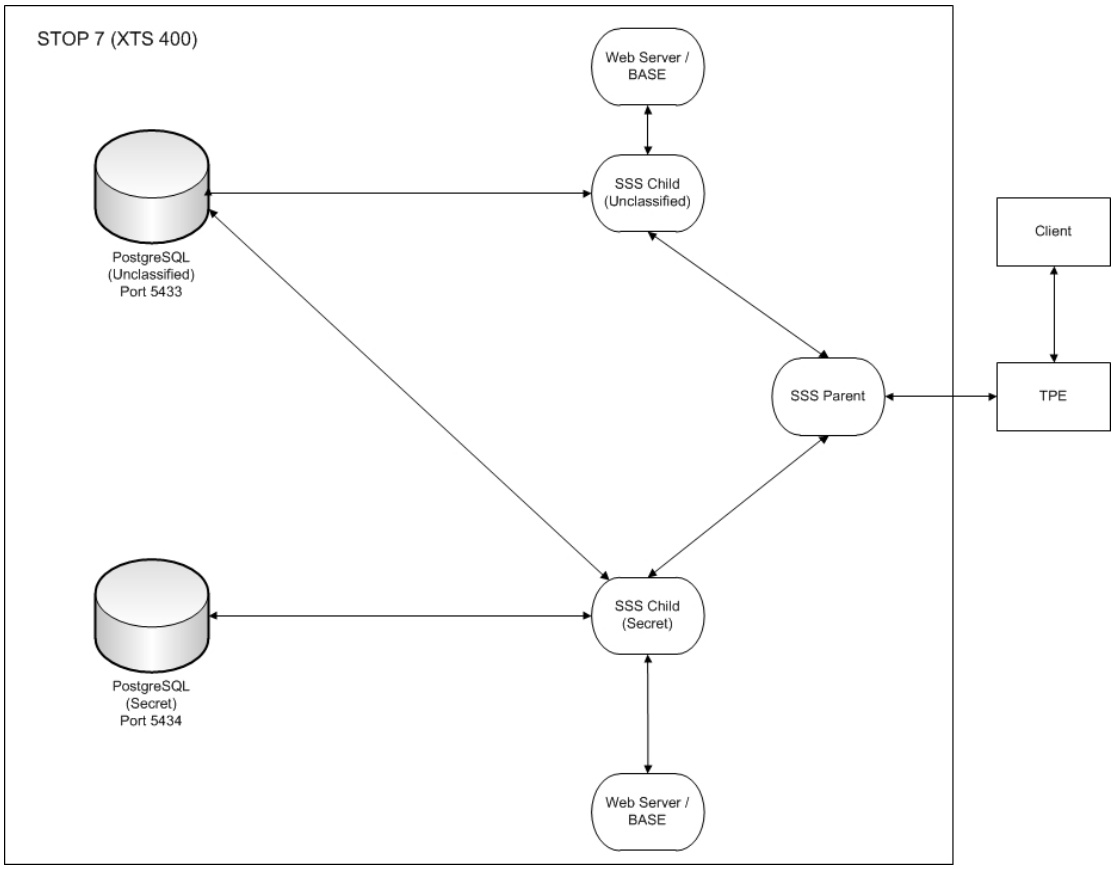

Figure 17. BASE test setup

Test ID	Test Type	Action	Expected Result
D1	Connect equal	Login at SIM_SECRET and connect to the SIM_SECRET IDS database	Advisory label of SIM_SECRET is displayed. The SIM_SECRET IDS data is displayed on the main page.
D2	Connect down	Connect to the SIM_UNCLASSIFIED IDS database	Advisory label of SECRET is still displayed. Only the "UNCLASSIFIED" checkbox is checked. An error message showing that it cannot connect to the UNCLASSIFIED database is displayed.
D3	Incorrect parameters	Set the IP address and port number of database to connect to values not belonging to PGPool-II or IDS database	An error message showing that it cannot connect to the database is displayed.

Table 6. Test cases for BASE without proxy

C. DEVELOPMENTAL TESTING RESULTS

Developmental testing of PGPool-II did not reveal any unforeseen problems. It was able to allow connections from clients with session levels that dominate the security level of the IDS databases and deny those that do not dominate. For clients with session levels that dominate the security level of the IDS databases, PGPool-II was able to allow read access. PGPool-II was able to deny write access to clients whose session levels are not equal to the security level of IDS databases. For the exception testing, PGPool-II was also able to deny access to unauthorized applications such as *psql*.

Developmental testing of BASE was also completed successfully. The advisory label of the current session level is always displayed. The checkboxes displayed levels that are dominated by the current session level. Advisory labels are also displayed at the start and end of each IDS data block, indicating the security level of the data displayed in that section. The tests also demonstrate that BASE is unable to connect to an IDS database with a lower security level if it configured to connect to the IDS databases directly.

D. ACCEPTANCE TESTING

The goal of acceptance testing is to verify that the implemented PGPool-II and BASE application are able display “live” IDS alerts generated by IDS machines using attack traffic from the simulated attacker machine. This updated data should be presented when the web page is refreshed. Multiple clients should also be able to access the different IDS databases at the same time. The test cases are presented in Table 7.

Test ID	Test Type	Action	Expected Result
E1	Updates to SIM_UNCLASSIFIED IDS database while logged in at session level of SIM_SECRET	Refresh web page	The SIM_UNCLASSIFIED section should display the IDS data that is not updated.
E2	Updates to SIM_SECRET IDS database while logged in at session level of SIM_SECRET	Refresh web page	The SIM_SECRET section should display the updated IDS data.
E3	Updates to SIM_UNCLASSIFIED	Refresh web page	The SIM_UNCLASSIFIED section should display the

	IDS database while logged in at session level of SIM_UNCLASSIFIED		updated IDS data.
E4	View the now updated SIM_UNCLASSIFIED data while logged in at session level of SIM_SECRET	Login. Ensure that the SIM_UNCLASSIFIED checkbox is checked and click on the “Go” button.	The SIM_UNCLASSIFIED section should now display the updated IDS data.
E3	Multiple clients access	Login in multiple clients at different security levels	The web page of each client should display data relevant to its security level.

Table 7. Acceptance test cases

E. ACCEPTANCE TEST RESULTS

The results of the acceptance tests were successful, demonstrating that updated data can be displayed when an IDS generates new alerts. Multiple clients at different security levels can also access the different IDS databases at the same time. Together with the developmental tests, the acceptance tests have shown that the current implementation of PGPool-II and BASE meets the top-level requirements described in Chapter III, Section D. Detailed test results are provided in Appendix D.

F. SUMMARY

This chapter described the developmental tests performed on the PGPool-II and BASE that were implemented and modified for this project, and the acceptance tests performed to ensure that the PGPool-II and BASE were able to interact within the MYSEA context to fulfill the top level requirements of this project. All the tests conducted were successful and the test outputs are provided in Appendix D. The next chapter concludes with a project summary and suggestions for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION

This chapter covers some of the future work that could be done to improve the IDS and database proxy, and the conclusion for this thesis.

A. FUTURE WORK

Several issues arose from this work that suggest further study and research to resolve them.

1. PGPool-II

Whenever a connection request from the client is sent to PGPool-II, it will contact the Remote Connection database to perform a check to determine if the client's session level is higher than or equal to the security level of the IDS database. If the client's session level is higher than or equal to the security level of the IDS database, PGPool-II will forward the connection request to the IDS database. This connection request will always be performed before the BASE application issues a query. This gives rise to a potential covert channel using the connection requests for lower level classification databases. A malicious program could be used to cause the BASE application at a higher security level to connect or disconnect to a lower level IDS database at timed intervals. Another malicious program running at that lower level, could probe the database at timed intervals to determine if a connection between PGPool-II and the database exists. The malicious program running at that lower level could then extract some information based on the connection status, resulting in a timing covert channel. Further analysis should also be conducted to determine if storage channels exist.

A possible solution to this is to maintain a continuous connection between PGPool-II and the corresponding IDS database. As PGPool-II will always be connected to the corresponding IDS database, the connections from the BASE application to the IDS database through PGPool-II cannot be used as a timing covert channel. This would eliminate the problem of a covert channel in this case.

2. BASE Event Cache

In the current implementation, the automatic update to the event cache of the BASE-specific databases with the security levels lower than that of the user's current session level is disabled, as it causes a premature termination of the display due to the error messages returned when attempting to perform a write-down. As a result, the user is only able to view the updated data from the event cache of the BASE-specific database with the security level equal to that of the user's current session level. He is only able to view the older data for the databases with the security level lower than that of the current session level. In order to update the event cache, the user has to log out of the current session level and log in at the session level where the update is required. He then has to log out and log in again at the previous session level to view the updated consolidated IDS data.

This issue arose because of the way BASE is implemented. It redundantly stores commonly used information in its own table to reduce the processing time required to retrieve it from several Snort tables. A possible solution is to modify BASE to write the event cache of all databases that BASE can query to the existing table, *acid_event*, in the database running at the level equal to the current session level. This can be done by adding an additional field as shown in Table 8 to indicate the security level of that event.

Security Level	**BASE Event Table Information**
SIM_UNCLASSIFIED	Alert 1 information
SIM_UNCLASSIFIED	Alert 2 information
SIM_SECRET	Alert 3 information
SIM_SECRET	Alert 4 information

Table 8. Additional field to BASE event cache table

Another possible solution is to implement addition event cache tables for each security level dominated by the security level of the database. However, this is not as scalable as the previous solution as additional tables have to be added at various security levels if the security level list grows.

3. BASE Advisory Labels

The set of advisory labels used by the BASE application is currently maintained in the BASE configuration files. The MYSEA server also maintains a separate set of advisory labels. If the label list in the MYSEA server grows, or if other labels for integrity or compartments are added, the set of advisory labels in the BASE configuration files would have to be updated to reflect these changes. Inconsistencies may arise if the set of advisory labels for the MYSEA server and BASE are not synchronized. Instead of having to maintain a separate set of advisory labels in the BASE configuration files, the BASE application could be modified to extract them from the MYSEA server. This would prevent inconsistencies in the label sets between BASE and the MYSEA server. Changes to the label set in the MYSEA server may be required in order to provide the mapping between the label and the corresponding binary representation.

4. IDS Improvements

Snort is the current choice of IDS for the MYSEA environment. However, Snort is single-threaded and thus, is able to look at only one packet at a time. Suricata, on the other hand, is multi-threaded, which allows it to concurrently inspect multiple network packets. This improves the chances of detecting attack traffic. While the current performance of Suricata is not up to par with Snort [10], development is ongoing to improve it. Further analysis should be done to determine if Suricata could replace Snort when its implementation becomes more mature.

B. CONCLUSION

The following questions were asked at the start of this thesis.

1. Can a trusted mechanism be designed that will permit operators to read intrusion detection information at multiple classification levels while enforcing the system's overall mandatory confidentiality policy?
2. The SQL database allows both read and write access when connected. Can the designed trusted mechanism restrict write access when a user at a higher security level connects to a database of a lower security level?

This thesis has addressed the two questions with the design and implementation of a database proxy and BASE modification. The database proxy mediates the access between the BASE application and the IDS databases to enforce the system's overall mandatory confidentiality policy. The database proxy prevents the BASE application from reading up and writing down while the modifications to the BASE application would allow operators to view intrusion detection information at multiple classification levels within a single web page.

The MYSEA project aims to provide a distributed multilevel secure operating environment that allows authenticated users to access data as permitted by the mandatory access control policy enforced by the MYSEA MLS server. With the design and implementation of the database proxy, it has been extended to provide "IDS read-down" capability to allow the system security analysts to obtain a more integrated view of the IDS alerts. This extension lays the groundwork to support real-time system response to network attacks.

APPENDIX A. SOURCE CODE LISTING

This appendix provides a listing of the BASE source code files that were created or modified for this project. The files that were modified for this project are indicated with an asterisk. All these files reside on the MYSEA server.

The following file generates the header information, which includes the advisory label, checkbox options as shown in Figure 13 in Chapter IV.

`base_mysea.php`

The following files were modified to include the BASE data presentation logic as illustrated in Figure 14 in Chapter IV.

`base_ag_main.php*`

`base_common.php*`

`base_conf.php*`

`base_main.php*`

`base_maintanence.php*`

`base_qry_alert.php*`

`base_qry_main.php*`

`base_stat_alerts.php*`

`base_stat_ipaddr.php*`

`base_stat_iplink.php*`

`base_stat_ports.php*`

`base_stat_sensor.php*`

`base_stat_time.php*`

`base_stat_uaddr.php*`

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. INSTALLATION PROCEDURES

This appendix outlines the installation procedures for installing PGPool-II on Fedora 12 and MYSEA, and BASE on MYSEA.

A. PGPOOL-II INSTALLATION ON FEDORA 12

This section covers the installation of PGPool-II and PostgreSQL on Fedora 12. PostgreSQL has to be installed first as PGPool-II requires a library file from PostgreSQL. The setup is as shown in Figure 18.

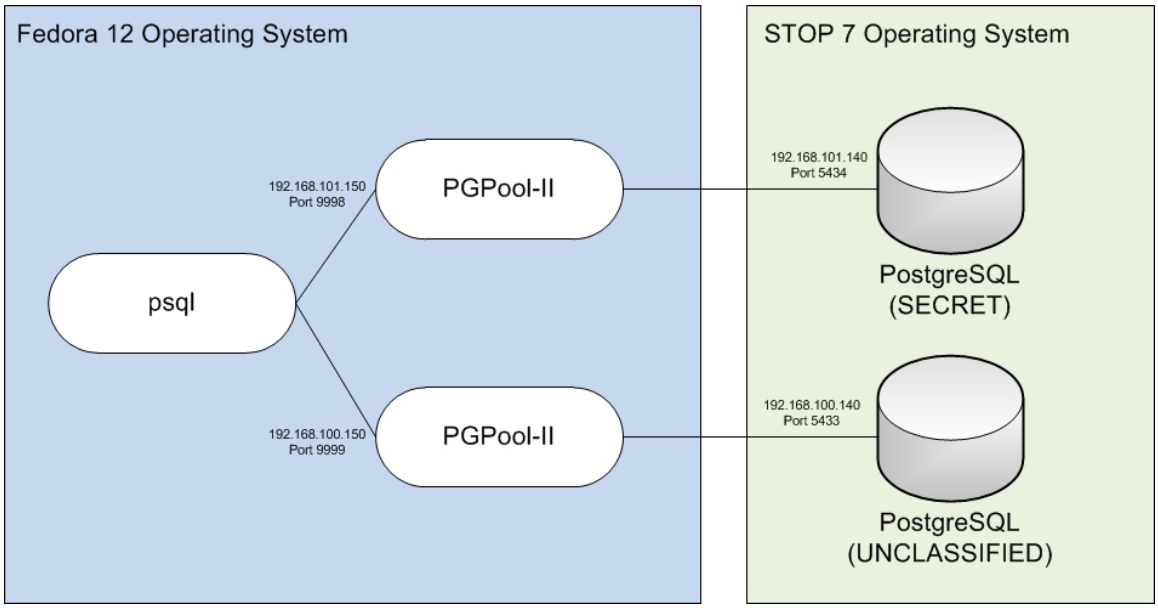

Figure 18. PGPool-II setup on Fedora 12

1. PostgreSQL Installation

1. Download PostgreSQL 8.4.4 from www.postgres.org to obtain the tar file postgresql-8.4.4.tar to the folder /home/Student/Downloads

2. Root access is required for the following steps. To get root access, first launch the Terminal and type the following:

   su

   Password: <*root password*>

3. Copy the tar file to /usr/local/src and install PostgreSQL by typing the following commands

```
cp /home/Student/Downloads/postgresql-8.4.4.tar
/usr/local/src

cd /usr/local/src

tar –xvf postgresql-8.4.4.tar

cd postgresql-8.4.4

./configure

make

make install
```

4. As only the library files from PostgreSQL are required, no further configuration is necessary and the installation for PostgreSQL is complete.

2. PGPool-II Installation

1. Download PGPool-II 2.3.3 from pgfoundry.org/projects/pgpool/ to obtain the tar file pgpool-II-2.3.3.tar to the folder /home/Student/Downloads

2. At the same Terminal, copy the tar file to /usr/local/src and install PGPool-II by typing the following commands

```
cp /home/Student/Downloads/pgpool-II.2.3.3.tar
/usr/local/src

cd /usr/local/src

tar –xvf pgpool-II.2.3.3.tar

cd pgpool-II.2.3.3

./configure

make
```

make install

3. As the test setup requires two instances of PGPool-II, two copies of the configuration files are required.

cp /usr/local/etc/pgpool.conf.sample /usr/local/etc/pgpool_unclassified.conf

cp /usr/local/etc/pgpool.conf.sample /usr/local/etc/pgpool_secret.conf

4. Configure the pgpool_unclassified.conf located at /usr/local/etc/ pgpool_unclassified.conf. Configure the parameters so that they look like the following:

```
listen_addresses = '*'

port = 9999

pcp_port = 9898

pid_file_name='/var/run/pgpool/pgpool_unclassified.pid'

backend_hostname0 = '192.168.100.140'

backend_port0 = 5433
```

5. Configure the pgpool_secret.conf located at /usr/local/etc/ pgpool_secret.conf. Configure the parameters so that they look like the following:

```
listen_addresses = '*'

port = 9998

pcp_port = 9897

pid_file_name='/var/run/pgpool/pgpool_secret.pid'

backend_hostname0 = '192.168.101.140'

backend_port0 = 5434
```

6. Start the two instances of PGPool-II with the following commands

pgpool –f /usr/local/etc/pgpool_unclassified.conf &

pgpool –f /usr/local/etc/pgpool_secret.conf &

7. To verify that both instances are started, type

ps –ef | grep pgpool

There should be two entries in the output, which correspond to:

pgpool –f /usr/local/etc/pgpool_unclassified.conf and

pgpool –f /usr/local/etc/pgpool_secret.conf

To test the connection to the IDS databases, start up *psql* in /usr/local/pgsql/bin

cd /usr/local/pgsql/bin

To connect to the SIM_UNCLASSIFIED IDS database

./psql –h localhost –p 9999 –U snort

Password for user snort: *<snort password>*

At the display prompt, type

snort=> *select * from acid_event;*

The results of the query should be displayed. Note the number of rows then quit psql.

snort=> \q

Next connect to the SIM_SECRET IDS database

./psql –h localhost –p 9998 –U snort

Password for user snort: *<snort password>*

At the display prompt, type

snort=> *select * from acid_event;*

The results of the query should be displayed. Note the number of rows. The number of rows should be different. This proves that both instances of PGPool-II are able to connect to the two different IDS databases.

Quit the psql application by typing

snort=> *\q*

B. PGPOOL-II AND BASE INSTALLATION ON STOP 7

This section covers installation of the modified PGPool-II and BASE on MYSEA running the STOP 7 OS. The IDS databases are assumed to be running already.

1. MYSEA Server Setup

The general steps to set up the MYSEA server are as follows:

1. Get all the source code from the MYSEA Configuration Management archive.
2. Perform the standard procedures for MYSEA installation and customize the network settings.
3. Customize the MYSEA server with PostgreSQL installation, PGPool-II installation, PHP (with PostgreSQL support) and backend port configurations to support the database proxy.

The detailed installation procedures can be obtained from "MYSEA Database Proxy Prototype Installation Manual, Version 1.0, November 2010" [16].

2. BASE Installation

This section assumes that no MYSEA server processes are currently running. The steps to install the BASE application on the MYSEA server is as follows:

1. Remove the Default Route.

   a. Login as ‘`madmin`’ user with password ‘`<madmin password>`’

   b. Get the required privilege for making changes

   ```
   sec_label –p –l admin,all_exempt
   ```

   c. Modify MYSEA route file (comment out default route)

   ```
   vi /xts/etc/startup.d/20-routes
   ```

   Modify the line

   `route add 0.0.0.0/0 192.168.0.254` to

   `#route add 0.0.0.0/0 192.168.0.254`

   e. Reboot the system for the changes to take effect

   ```
   shutdown -r
   ```

2. Copy the modified BASE Application into the MYSEA server. Obtain a CISR CM archive “Thesis-Ang-2010” CD containing the MYSEA_BASE.tar tarball, The following assumes that the CD device on the STOP 7.2.1 VM is on /dev/hdb.

   a. Login as ‘`madmin`’ user with password ‘`<madmin password>`’

   b. Get the required privilege for making changes

   ```
   sec_label –p –l admin,all_exempt,m_priv_macdac
   ```

   c. Mount the CD device.

   ```
   mount –r /dev/hdb /mnt/cdrom
   ```

   d. Copy MYSEA_BASE.tar to the /home/madmin/mysea directory.

```
cp /mnt/cdrom/MYSEA_BASE.tar /home/madmin/mysea
```

e. Copy the PGPool-II test configuration files to /local/mysea/conf directory.

```
cp /mnt/cdrom/pgpool*.wrong /local/mysea/conf
```

f. Modify the PGPool-II test configuration files permission.

```
sec_label –l m_user_obj:syslo:madmin.m_sys.0444
/local/mysea/conf/pgpool*.wrong
```

g. Unmount the CD device.

```
umount /mnt/cdrom
```

3. Install the modified BASE application.

a. Extract the modified BASE application into the /home/madmin/mysea/server/home_http_data/htdocs directory.

```
tar -C
/home/madmin/mysea/server/home_http_data/htdocs -xvf
/home/madmin/mysea/MYSEA_BASE.tar
```

b. Remove the ids_demo/setup directory:

```
rm -rf
/home/madmin/mysea/server/home_http_data/htdocs/ids_demo/se
tup
```

c. Go to /home/madmin/mysea/server directory and run the make_data_tar script.

```
cd /home/madmin/mysea/server
```

```
./make_data_tar
```

d. Go to /home/madmin/mysea/server/scripts directory and run the mysea_datainst.sh script.

```
cd /home/madmin/mysea/server/scripts
```

```
./mysea_datainst.sh
```

When prompted, enter '1' for TWiki Data Demo

APPENDIX C. TEST PROCEDURES

This appendix documents the test procedures used in the Test Plan presented in Chapter V. The test network should be set up as shown in Figure 19 before testing.

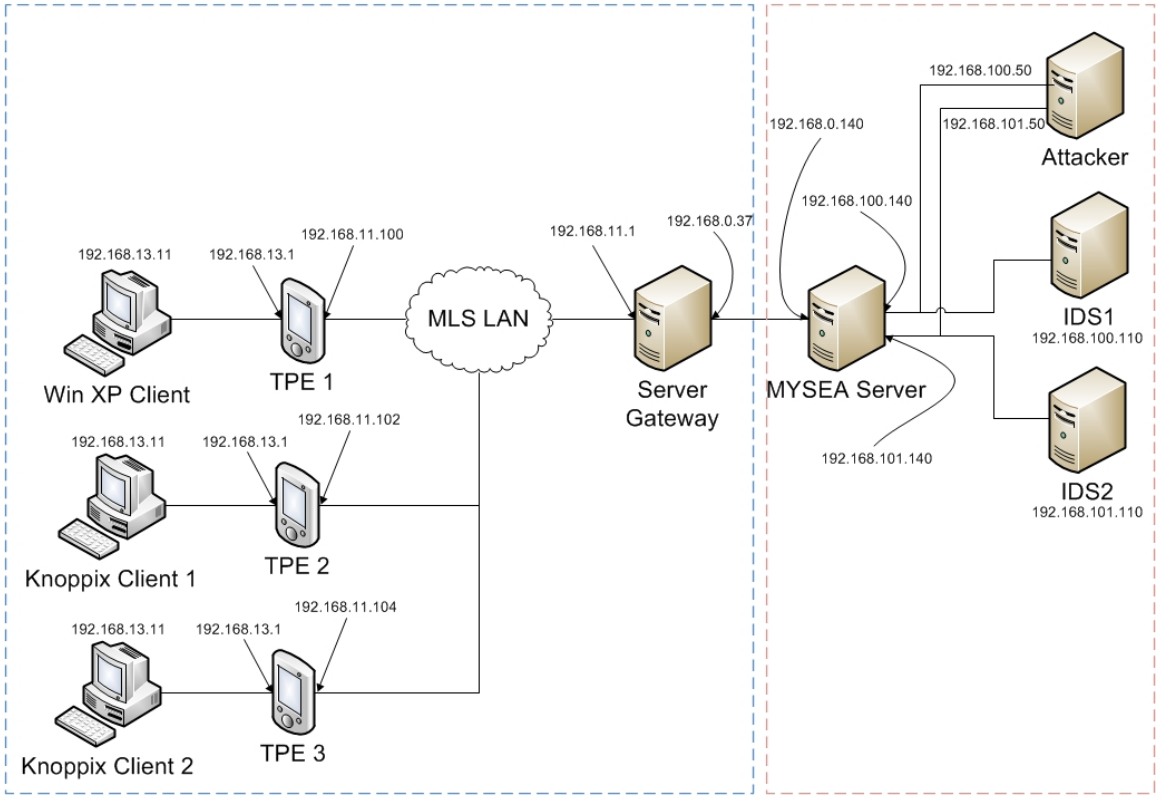

Figure 19. Detailed test setup

The server gateway, TPEs and the clients (in the left dotted box) are part of the standard MYSEA setup and thus, their setup is not discussed here. The setup of the IDS and attacker machines can be found in Appendix B of Tenhunen's thesis [3]. For the setup of the MYSEA server, refer to Section B of Appendix B.

As the numbers of machines are limited, Virtual Machines (VMs) are used in the place of actual machines. The following steps are then performed.

1. Power up all the systems in the test network.

2. Start MYSEA Server VM (MYSEA-Server-IDS-STOP 7.2.1)

   Note: <SAK> is <Alt> + <SysRq/PrintScreen>

   a. <SAK>

   b. Login as ‘`madmin`’, password is ‘‘`<madmin password>`’

   c. Go to ‘/local/mysea/scripts’

   `cd /local/mysea/scripts`

   d. Start MYSEA daemons

   `./mysea_start_daemon.sh 8000`

   e. Start the PostgreSQL databases

   `./start_postgres.sh SIM_UNCLASSIFIED`

   `./start_postgres.sh SIM_SECRET`

   f. Start PGPool-II

   `./start_pgpool.sh`

3. Start Snort on IDS1 (Debian4.0 – IDS1 VM)

   a. Login as ‘`root`’, password is ‘‘`<root password>`’

   b. Start snort:

   `snort -u snort -g snort -c /etc/snort/snort.conf &> ids1_test1.txt`

4. Start Snort on IDS2 (Debian4.0 – IDS VM)

   a. Login as ‘`root`’, password is ‘‘`<root password>`’

   b. Start snort:

   `snort -u snort -g snort -c /etc/snort/snort.conf &> ids2_test1.txt`

5. Start the IDS Wakeup: Debian4.0 – idswakeup VM

a. Login in with username ‘mdemo1’ and password ‘<mdemo1 password>’.

b. Start the ‘root’ terminal located on the desktop and enter ‘<root password>’ as the password.

6. Connect the Server Gateway VM to MYSEA Server

   a. Login as ‘root’, password is ‘‘<root password>’

   b. Open a terminal window (located on desktop)

   c. Change directory to /root/mysea/bin

   cd /root/mysea/bin

   d. Restart DSS configuration

   ./dss_restart

   e. Register DSS Client:

   ./dss_client 192.168.0.140

7. Connect the TPE1 VM

   a. Login as ‘root’, password is ‘‘<root password>’

   b. Open a terminal window (located on desktop)

   c. Change directory to /root/mysea/bin

   cd /root/mysea/bin

   d. Restart DSS configuration

   ./dss_restart

   e. Register DSS Client:

   ./dss_client 192.168.0.140

   f. Open another terminal window and type

   cd /root/mysea/bin

g. Start the TPE

./tcbe.py

h. Click on the ‘SAR’ button

i. Enter ‘mdemo1’ as the user name

j. Enter ‘‘<mdemo1 password>’ as the password

k. Click on the ‘SAR’ button. Enter the command ‘sl’

Enter session level as ‘SIM_SECRET’

l. Click on ‘SAR’ button. Enter the command ‘run’. Wait for the “RUN command completed” message to be displayed.

8. On the Windows XP system,

a. Launch the web browser (Internet Explorer)

b. Browse default MYSEA Server page at

http://mlsserver.cisrlabmlstestbed1.com

The advisory label displayed should be ‘SIM_SECRET’.

c. Browse to BASE login page

http://mlsserver.cisrlabmlstestbed1.com/ids_demo/index.php

A. PGPOOL-II TEST PROCEDURES

At the STOP 7 prompt, navigate to /home/http/htdocs/ids_demo.

cd /home/http/htdocs/ids_demo

The test procedures are as follows:

1. For test cases A1 and A2, use the configuration file stored in the directory navigated to by doing:

cp base_conf.php.a-equal base_conf.php

2. On the Windows XP system, refresh the BASE login page on the web browser.

3. Perform the actions listed for test cases A1 and A2

4. On the STOP 7 OS, use the base_conf.php.a-equal2 file. This file sets the '`$event_cache_auto_update'` parameter to 1, which causes BASE to perform a write query to the IDS database.

   `cp base_conf.php.a-equal2 base_conf.php`

5. Generate traffic on the IDS Wakeup: Debian4.0 – idswakeup VM

   a. Generate alert traffic for the SIM_UNCLASSIFIED network

   `idswakeup 192.168.100.50 192.168.100.140`

   b. Generate alert traffic for the SIM_SECRET network

   `idswakeup 192.168.101.50 192.168.101.140`

6. On the Windows XP VM, browse to the BASE login page again and then perform the actions specified in A3.

7. On the STOP 7 VM, overwrite the 'base_conf.php' file with the 'base_conf.php.a-readdown' file.

   `cp base_conf.php.a-readdown base_conf.php`

8. On the Windows XP VM, browse to the BASE login page again. Login with username as '`snort`' and password as '`<snort password>`'. Ensure that the SIM_UNCLASSIFIED checkbox is checked, and the SIM_SECRET checkbox is not and click on the 'Go' button.

9. Perform the actions specified in A4 and A5.

10. Repeat step 7, followed by clicking on the "Go" button. Click on the "Cache and Status" link at the bottom of the web page.

11. Perform the actions specified in A6.

12. Close the web browser on the Windows XP machine.

13. On the STOP 7 VM, overwrite the ‘base_conf.php’ file with the ‘base_conf.php.connectup’ file.

    ```
    cp base_conf.php.connectup base_conf.php
    ```

14. On the TPE1 VM, click on the ‘SAR’ button in the TPE window and enter the command ‘`sl`’. Enter ‘`SIM_UNCLASSIFIED`’ for the session level.

15. Click the ‘SAR’ button again and enter the command ‘`run`’. Wait for the “Run command completed” message to be displayed.

16. On the Windows XP VM, launch the Internet Explorer web browser.

17. Browse to the MYSEA Server page at http://mlsserver.cisrlabmlstestbed1.com. The advisory label displayed should be ‘SIM_UNCLASSIFIED’.

18. Browse to the BASE login page. The advisory label displayed should be ‘SIM_UNCLASSIFIED’. However, the underlying IP address and port number has been deliberately mis-configured to connect to the SIM_SECRET IDS database.

19. Perform the actions specified in A7.

20. On the STOP 7 VM, replace the ‘base_conf.php’ to the MYSEA settings.

    ```
    cp base_conf.php.mysea base_conf.php
    ```

21. Close the web browser on the Windows XP VM.

22. On the TPE1 VM, click on the ‘SAR’ button in the TPE window and enter the command ‘`sl`’. Enter ‘`SIM_SECRET`’ for the session level.

23. Click the ‘SAR’ button again and enter the command ‘`run`’. Wait for the “Run command completed” message to be displayed.

24. On the STOP 7 VM, terminate PGPool-II.

/local/mysea/scripts/stop_pgpool.sh

25. Save the original configuration files first.

cp /local/mysea/conf/pgpool_SIM_UNCLASSIFIED.conf /local/mysea/conf/pgpool_SIM_UNCLASSIFIED.conf.orig

cp /local/mysea/conf/pgpool_SIM_SECRET.conf /local/mysea/conf /pgpool_SIM_SECRET.conf.orig

26. Overwrite the PGPool-II configuration with files that are misconfigured.

sec_label –p –l admin,all_exempt:

cp /local/mysea/conf/pgpool_SIM_UNCLASSIFIED.conf.wrong /local/mysea/conf/pgpool_SIM_UNCLASSIFIED.conf

cp /local/mysea/conf/pgpool_SIM_SECRET.conf.wrong /local/mysea/conf /pgpool_SIM_SECRET.conf

sec_label –p –l admin:

27. Restart PGPool-II.

/local/mysea/scripts/start_pgpool.sh

28. On the Windows XP VM, launch the web browser.

29. Browse to the BASE login page and perform test A8.

30. On the STOP 7 VM, restore the configuration files to their original settings.

sec_label –p –l admin,all_exempt:

cp /local/mysea/conf/pgpool_SIM_UNCLASSIFIED.conf.orig /local/mysea/conf/pgpool_SIM_UNCLASSIFIED.conf

cp /local/mysea/conf/pgpool_SIM_SECRET.conf.orig /local/mysea/conf /pgpool_SIM_SECRET.conf

sec_label –p –l admin:

31. Stop and restart PGPool-II.

/local/mysea/scripts/stop_pgpool.sh

/local/mysea/scripts/start_pgpool.sh

32. Close the web browser on the Windows XP VM.

Test ID	Test Type	Action	Expected Result
A1	Connect equal	Enter the username as 'snort' and password as ''<snort password>'	Connection successful. The main page will be displayed.
A2	Read equal	This step continues from A1. No action required	Main page is displayed without errors
A3	Write equal	Enter the username as 'snort' and password as ''<snort password>'	Main page is displayed without errors.
A4	Connect down	Click on the "Go" button.	Connection successful. The main page will be displayed.
A5	Read down	This step continues from A1. No action required	Main page is displayed without errors
A6	Write down	Click on "Update Alert Cache" under "Alert Information Cache" in the SIM_UNCLASSIFIED section.	Main page is displayed with error message: "Database ERROR: server closed the connection unexpectedly."
A7	Connect up	Enter the username as 'snort' and password as	Connection unsuccessful. The following error

		‘<snort password>’	message is displayed “Error (p) connecting to DB: snort@192.168.0.140:9998”
A8	Incorrect parameters	Enter the username as ‘snort’ and password as ‘<snort password>’	Connection unsuccessful. The following error message is displayed “Error (p) connecting to DB: snort@192.168.0.140:9998 (The error page may take a while to load)

For the exception test case, launch the Fedora 12 VM. Login with username ‘Student’ and password ‘<Student password>’. Open a terminal and navigate to /usr/local/pgsql/bin.

```
cd /usr/local/pgsql/bin
```

Perform the test case by typing the following in the prompt.

```
./psql –h 192.168.0.140 –p 9999 –U snort
```

and

```
./psql –h 192.168.0.140 –p 9998 –U snort
```

Both commands should return the result “psql: server closed the connection abnormally before or while processing the request.

B. BASE TEST PROCEDURES

The test procedures are as follows:

1. At the STOP 7 prompt, navigate to /home/http/htdocs/ids_demo.

   ```
   cd /home/http/htdocs/ids_demo
   ```

2. In the same directory, overwrite the ‘base_conf.php’ configuration file with ‘base_config.php.mysea’.

   ```
   cp base_conf.php.mysea base_conf.php
   ```

3. On the TPE1 VM, click on the ‘SAR’ button in the TPE window and enter the command ‘`sl`’. Enter ‘`SIM_UNCLASSIFIED`’ for the session level.

4. Click the ‘SAR’ button again and enter the command ‘`run`’. Wait for the “Run command completed” message to be displayed.

5. On the Windows XP VM, launch the Internet Explorer web browser.

6. Browse to the MYSEA Server page at http://mlsserver.cisrlabmlstestbed1.com. The advisory label displayed should be ‘SIM_UNCLASSIFIED’.

7. Browse to the BASE login page at:

   http://mlsserver.cisrlabmlstestbed1.com/ids_demo/index.php

8. Perform the actions listed in Test C1 to C3. Close the web browser after performing the actions.

9. On the TPE1 VM, click on the ‘SAR’ button in the TPE window and enter the command ‘`sl`’. Enter ‘`SIM_SECRET`’ for the session level.

10. Click the ‘SAR’ button again and enter the command ‘`run`’. Wait for the “Run command completed” message to be displayed.

11. On the Windows XP VM, launch the Internet Explorer web browser.

12. Browse to the MYSEA Server page at http://mlsserver.cisrlabmlstestbed1.com. The advisory label displayed should be ‘SIM_SECRET’.

13. Browse to the BASE login page at:

    http://mlsserver.cisrlabmlstestbed1.com/ids_demo/index.php

14. Perform the actions listed in Test C4 to C7.

15. Ensure that both the SIM_UNCLASSIFIED and SIM_SECRET checkboxes are checked and click on the “Go” button. Perform action listed in Test C8.

16. Click on the ‘Home’ link. Perform action listed in Test C9.

17. Close the web browser after performing the actions.

Test ID	**Test Type**	**Action**	**Expected Result**
C1	Login Display at SIM_UNCLASSIFIED level	No action required	Advisory label of SIM_UNCLASSIFIED is displayed. There is also only one option displayed for security level selection.
C2	Login at SIM_UNCLASSIFIED level	Login with username ‘`snort`’ and password ‘`<snort password>`’	Login Successfully. The main page is displayed with the advisory label of SIM_UNCLASSIFIED. There is also only one option displayed for security level selection. Advisory labels are placed at the start and end of the displayed IDS data section.
C3	Page Navigation for SIM_UNCLASSIFIED	Click on a link on the page.	A new page is displayed on the same Window, showing the query results of the SIM_UNCLASSIFIED database. Advisory

			labels are placed at the start and end of the IDS data section
C4	Login Display at SIM_SECRET level	No action required	Advisory label of SIM_SECRET is displayed. There are also only two options (SIM_UNCLASSIFIED and SIM_SECRET) displayed for security level selection.
C5	Login at SIM_SECRET	Login with username '`snort`' and password '`<snort password>`'	Login Successfully. Main page is displayed. Advisory Label of SIM_SECRET is displayed
C6	Data consolidation	Ensure that both the SIM_UNCLASSIFIED and SIM_SECRET checkboxes are checked and click on the "Go" button.	Both checkboxes remain checked. Advisory labels for SIM_UNCLASSIFIED and SIM_SECRET data are placed at the start and end of each IDS data section.
C7	Lower level data only	Ensure that only the UNCLASSIFIED checkbox is checked and click on the "Go" button.	The SIM_UNCLASSIFIED checkbox remains checked. Only the SIM_UNCLASSIFIED data is displayed and Advisory labels for SIM_UNCLASSIFIED data are placed at the start and end of the

			IDS data section.
C8	Page Navigation at SIM_SECRET level	Click on a link in the SIM_SECRET section of the data.	The resulting page will be displayed in the same window, showing only the SIM_SECRET data.
C9	Page Navigation at SIM_UNCLASSIFIED	Click on a link in the SIM_UNCLASSIFIED section of the data.	The resulting page will be displayed in the same window, showing only the SIM_UNCLASSIFIED data.

The test procedures for the second set of tests are as follows:

1. At the STOP 7 prompt, navigate to /home/http/htdocs/ids_demo.

   cd /home/http/htdocs/ids_demo

2. In the same directory, overwrite the 'base_conf.php' configuration file with 'base_config.php.direct'. The IP addresses and port numbers are configured to connect to the IDS databases directly.

   cp base_conf.php.direct base_conf.php

3. On the TPE1 VM, click on the 'SAR' button in the TPE window and enter the command '`sl`'. Enter '`SIM_SECRET`' for the session level.

4. Click the 'SAR' button again and enter the command '`run`'. Wait for the "Run command completed" message to be displayed.

5. On the Windows XP VM, launch the Internet Explorer web browser.

6. Browse to the MYSEA Server page at http://mlsserver.cisrlabmlstestbed1.com. The advisory label displayed should be 'SIM_SECRET'.

7. Browse to the BASE login page at:

   http://mlsserver.cisrlabmlstestbed1.com/ids_demo/index.php

8. Perform the actions listed in Tests D1 and D2.

9. On the STOP 7 VM, replace the 'base_conf.php' with 'base_conf.php.wrong'.

   ```
   cp base_conf.php.wrong base_conf.php
   ```

10. On the Windows XP VM, return to the BASE login page. Perform the action list in Test D3. Close the web browser when done.

11. On the STOP 7 VM, restore the 'base_conf.php' to the original settings.

    ```
    cp base_conf.php.mysea base_conf.php
    ```

Test ID	**Test Type**	**Action**	**Expected Result**
D1	Connect equal	Ensure that only the SIM_SECRET checkbox is checked. Login with username '`snort`' and password '`<snort password>`'	Advisory label of SIM_SECRET is displayed. The SIM_SECRET IDS data is displayed on the main page.
D2	Connect down	Uncheck the SIM_SECRET checkbox, check the SIM_UNCLASSIFIED checkbox, and click on the "Go" button	Advisory label of SIM_SECRET is still displayed. An error message showing that it cannot connect to the SIM_UNCLASSIFIED database is displayed.

D3	Incorrect parameters	Login with username ‘snort’ and password ‘<snort password>’	An error message showing that it cannot connect to the database is displayed.

C. ACCEPTANCE TEST PROCEDURES

The test procedures for the acceptance tests are as follows:

1. Connect the TPE2 VM

   a. Login as ‘root’, password is ‘<root password>’

   b. Open a terminal window (located on desktop)

   c. Change directory to /root/mysea/bin

   cd /root/mysea/bin

   d. Restart DSS configuration

   ./dss_restart

   e. Register DSS Client:

   ./dss_client 192.168.0.140

   f. Open another terminal window and type

   cd /root/mysea/bin

   g. Start the TPE

   ./tcbe.py

   h. Click on the ‘SAR’ button

   i. Enter ‘mdemo2’ as the user name

   j. Enter ‘<mdemo2 password>’ as the password

   k. Click on the ‘SAR’ button. Enter the command ‘sl’

Enter session level as ‘SIM_UNCLASSIFIED’

l. Click on ‘SAR’ button. Enter the command ‘run’. Wait for the “RUN command completed” message to be displayed.

2. Connect the TPE3 VM. The steps are the same as in Step 1 except at the following steps.

   i. Enter ‘cudemo’ as the user name

   j. Enter ‘<cudemo password>’ as the password

   k. Click on the ‘SAR’ button. Enter the command ‘sl’

   Enter session level as ‘SIM_SECRET’

3. On the Windows XP VM, launch the Internet Explorer web browser.
4. Browse to the MYSEA Server page at http://mlsserver.cisrlabmlstestbed1.com.
5. Browse to the BASE login page at:

   http://mlsserver.cisrlabmlstestbed1.com/ids_demo/index.php

6. Login with username ‘snort’ and password ‘<snort password>’.
7. Ensure that both the SIM_UNCLASSIFIED and SIM_SECRET buttons are checked and click on the “Go” button. Both sets of IDS data should be displayed on the web page.
8. Generate traffic on the IDS Wakeup: Debian4.0 – idswakeup VM

   a. Generate alert traffic for the SIM_UNCLASSIFIED network

   ```
   idswakeup 192.168.100.50 192.168.100.140
   ```

   b. On the web browser in the Windows XP VM, perform action listed in Test E1. Return to the BASE main web page when done.

   c. Generate alert traffic for the SIM_SECRET network

`idswakeup 192.168.101.50 192.168.101.140`

d. Perform action listed in Test E2.

9. Close the web browser on the Windows XP VM.
10. On the TPE1 VM,

    a. Click on the ‘SAR’ button. Enter the command ‘`sl`’

    Enter session level as ‘`SIM_UNCLASSIFIED`’

    b. Click on ‘SAR’ button. Enter the command ‘`run`’. Wait for the “RUN command completed” message to be displayed.
11. Repeat step 3 to 6, then repeat step 8a.
12. Perform action listed in Test E3. Close the browser when done.
13. Repeat step 10 to 11. Enter session level as ‘`SIM_SECRET`’ in step 10a. In step 11, repeat step 8c instead of 8a.
14. On the Windows XP VM, perform the action listed in Test E4.
15. On both the two Knoppix Client VMs, launch the Firefox web browser. Repeat Steps 4 to 6.
16. Perform action listed in Test E5.

Test ID	**Test Type**	**Action**	**Expected Result**
E1	Updates to SIM_UNCLASSIFIED IDS database while logged in at session level of SIM_SECRET	Refresh web page. To see if there are updates to the SIM_UNCLASSIFIED IDS database, click on the “Cache & Status” link at the bottom of the web page.	The SIM_UNCLASSIFIED section should display the IDS data that is not updated. When the “Cache & Status” link is clicked, the value for total number of events should be higher

			than that of the event cache.
E2	Updates to SIM_SECRET IDS database while logged in at session level of SIM_SECRET	Refresh web page	The SIM_SECRET section should display the updated IDS data.
E3	Updates to SIM_UNCLASSIFIED IDS database while logged in at session level of SIM_UNCLASSIFIED	Refresh web page	The SIM_UNCLASSIFIED section should display the updated IDS data.
E4	View the now updated SIM_UNCLASSIFIED data while logged in at session level of SIM_SECRET	Login. Ensure that the SIM_UNCLASSIFIED checkbox is checked and click on the “Go” button.	The SIM_UNCLASSIFIED section should now display the updated IDS data.
E5	Multiple clients access	View main web page	Client 1 (Windows XP) SIM_UNCLASSIFIED and SIM_SECRET data are displayed. Client 2 (Knoppix) Only SIM_UNCLASSIFIED data is displayed. Client 3 (Knoppix) Same as Client 1

APPENDIX D. TEST RESULTS

This appendix presents the results of the development and acceptance tests performed in Appendix C.

A. DEVELOPMENT TEST RESULTS

This section covers the development test results for PGPool-II and the BASE application.

1. PGPool-II Test Results

The test results for PGPool-II are as follows:

Test ID	Test Type	Expect Result	Result (Pass/Fail)
A1	Connect equal	Connection successful. The main page will be displayed.	Pass
A2	Read equal	Main page is displayed without errors	Pass
A3	Write equal	Main page is displayed without errors.	Pass
A4	Connect down	Connection successful. The main page will be displayed.	Pass
A5	Read down	Main page is displayed without errors	Pass
A6	Write down	Main page is displayed with error message: “Database ERROR: server closed the connection unexpectedly.”	Pass
A7	Connect up	Connection unsuccessful. The	Pass

		following error message is displayed “Error (p) connecting to DB: snort@192.168.0.140”	
A8	Incorrect parameters	Connection unsuccessful. The following error message is displayed “Error (p) connecting to DB: snort@192.168.0.140:9998 (The error page may take a while to load)	Pass

The result for the exception test case is as follows:

Test ID	**Test Type**	**Expected Result**	**Result (Pass/Fail)**
B1	Unauthorized connection	Connection unsuccessful	Pass

2. BASE Test Results

The results for the first test suite of BASE are as follows:

Test ID	**Test Type**	**Expected Result**	**Result (Pass/Fail)**
C1	Login Display at SIM_UNCLASSIFIED level	Advisory label of SIM_UNCLASSIFIED is displayed. There is also only one option displayed for security level selection.	Pass
C2	Login at SIM_UNCLASSIFIED level	Login Successfully. The main page is displayed with the advisory label of	Pass

		SIM_UNCLASSIFIED. There is also only one option displayed for security level selection. Advisory labels are placed at the start and end of the displayed IDS data section.	
C3	Page Navigation for SIM_UNCLASSIFIED	A new page is displayed on the same Window, showing the query results of the SIM_UNCLASSIFIED database. Advisory labels are placed at the start and end of the IDS data section	Pass
C4	Login Display at SIM_SECRET level	Advisory label of SIM_SECRET is displayed. There are also only two options (SIM_UNCLASSIFIED and SIM_SECRET) displayed for security level selection.	Pass
C5	Login at SIM_SECRET	Login Successfully. Main page is displayed. Advisory Label of SIM_SECRET is displayed	Pass
C6	Data consolidation	Both checkboxes remain checked. Advisory labels for SIM_UNCLASSIFIED and SIM_SECRET data are placed at the start and end of each IDS data section.	Pass
C7	Lower level data only	The SIM_UNCLASSIFIED checkbox remains checked. Only the	Pass

		SIM_UNCLASSIFIED data is displayed and Advisory labels for SIM_UNCLASSIFIED data are placed at the start and end of the IDS data section.	
C8	Page Navigation at SIM_SECRET level	The resulting page will be displayed in the same window, showing only the SIM_SECRET data.	Pass
C9	Page Navigation at SIM_UNCLASSIFIED	The resulting page will be displayed in the same window, showing only the SIM_UNCLASSIFIED data.	Pass

The results for the second test suite of BASE are as follows:

Test ID	**Test Type**	**Expected Result**	**Result (Pass/Fail)**
D1	Connect equal	Advisory label of SIM_SECRET is displayed. The SIM_SECRET IDS data is displayed on the main page.	Pass
D2	Connect down	Advisory label of SIM_SECRET is still displayed. An error message showing that it cannot connect to the SIM_UNCLASSIFIED database is displayed.	Pass
D3	Incorrect parameters	An error message showing that it cannot connect to the database is displayed.	Pass

B. ACCEPTANCE TEST RESULTS

The acceptance test results are as follows.

Test ID	Test Type	Expected Result	Result (Pass/Fail)
E1	Updates to SIM_UNCLASSIFIED IDS database while logged in at session level of SIM_SECRET	The SIM_UNCLASSIFIED section should display the IDS data that is not updated. When the “Cache & Status” link is clicked, the value for total events should be higher than that of the event cache.	Pass
E2	Updates to SIM_SECRET IDS database while logged in at session level of SIM_SECRET	The SIM_SECRET section should display the updated IDS data.	Pass
	Updates to SIM_UNCLASSIFIED IDS database while logged in at session level of SIM_UNCLASSIFIED	The SIM_UNCLASSIFIED section should display the updated IDS data.	Pass
	View the now updated SIM_UNCLASSIFIED data while logged in at session level of SIM_SECRET	The SIM_UNCLASSIFIED section should now display the updated IDS data.	Pass
E3	Multiple clients access	Client 1 (Windows XP) SIM_UNCLASSIFIED and SIM_SECRET data are displayed.	Pass

		Client 2 (Knoppix) Only SIM_UNCLASSIFIED data is displayed. Client 3 (Knoppix) Same as Client 1	

LIST OF REFERENCES

[1] D. E. Bell and L. LaPadula, “Secure computer system: unified exposition and Multics interpretation,” *Technical Report ESD-TR-75-306*, Hanscom AFB, MA: The MITRE Corporation, 1975.

[2] C. E. Irvine, T. D. Nguyen, D. J. Shifflett, T. E. Levin, J. Khosalim, C. Prince, P. C. Clark, and M. Gondree, "MYSEA: The Monterey Security Architecture" in proc. *Workshop on Scalable Trusted Computing (ACM STC), Conference on Computer and Communications Security (CCS), Association for Computing Machinery (ACM)*, 2009.

[3] T. Tenhunen, “Implementing an Intrusion Detection System in the MYSEA architecture,” Master’s thesis, Naval Postgraduate School, June 2008.

[4] BAE Systems, *Information Assurance XTS-400 Trusted Computer Guard Datasheet,* July 2010.

[5] J. Horn, “IPSec-based dynamic security services for the MYSEA environment,” Master’s thesis, Naval Postgraduate School, June 2005.

[6] K. L. Ong, T. D. Nguyen, and C. E. Irvine, "Implementation of a multilevel Wiki for cross-domain collaboration," *3rd International Conference on Information Warfare and Security (ICIW 2008)*, Omaha, Nebraska, USA, pp. 293–304, April 2008.

[7] Snort (2010, May). Available: http://www.snort.org (Accessed: May 2010).

[8] T. Kohlenberg, et. al., *Snort IDS and IPS Toolkit, 2007*, Burlington, MA: Syngress Publishing, Inc.

[9] Sourcefire White Paper, “Snort threat components,” 2007. Available: http://www.sourcefire.com/resources/downloads/secured/snort_threat_components.pdf, (Accessed: May 2010).

[10] E. Messmer, “Is open source Snort dead? Depends who you ask,” Available: http://www.networkworld.com/news/2010/072010-is-snort-dead.html?t51hb (Accessed July 2010).

[11] Basic Analysis Security Engine (2009, May). Available: http://base.secureideas.net (Accessed May 2010).

[12] BASE 2 (2010, Aug). Available: http://base-ids-gui.sourceforge.net/, (Accessed October 2010).

[13] PGPool II, Available: http://pgpool.projects.postgresql.org (Accessed August 2010).

[14] SQL Relay, Available: http://sqlrelay.sourceforge.net/ (Accessed: August 2010).

[15] PL/Proxy, Available: http://pgfoundry.org/projects/plproxy/ (Accessed August 2010).

[16] MYSEA Database Proxy Prototype Installation Manual, Version 1.0, November 2010.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Kris Britton
   National Security Agency
   Fort Meade, MD

4. John Campbell
   National Security Agency
   Fort Meade, MD

5. Deborah Cooper
   DC Associates, LLC
   Reston, VA

6. Grace Crowder
   NSA
   Fort Meade, MD

7. Louise Davidson
   National Geospatial Agency
   Bethesda, MD

8. Vincent J. DiMaria
   National Security Agency
   Fort Meade, MD

9. Rob Dobry
   NSA
   Fort Meade, MD

10. Jennifer Guild
    SPAWAR
    Charleston, SC

11. CDR Scott Heller
SPAWAR
Charleston, SC

12. Dr. Steven King
ODUSD
Washington, DC

13. Steve LaFountain
NSA
Fort Meade, MD

14. Dr. Greg Larson
IDA
Alexandria, VA

15. Dr. Carl Landwehr
National Science Foundation
Arlington, VA

16. John Mildner
SPAWAR
Charleston, SC

17. Dr. Victor Piotrowski
National Science Foundation
Arlington, VA

18. Jim Roberts
Central Intelligence Agency
Reston, VA

19. Ed Schneider
IDA
Alexandria, VA

20. Mark Schneider
NSA
Fort Meade, MD

21. Keith Schwalm
Good Harbor Consulting, LLC
Washington, DC

22. Ken Shotting
NSA
Fort Meade, MD

23. Dr. Ralph Wachter
ONR
Arlington, VA

24. Dr. Cynthia E. Irvine
Naval Postgraduate School
Monterey, CA

25. Thuy D. Nguyen
Naval Postgraduate School
Monterey, CA

26. Dr. Yeo Tat Soon
National University of Singapore (NUS)
Singapore

27. Tan Lai Poh
National University of Singapore (NUS)
Singapore

28. Kah Kin Ang
Defence Science & Technology Agency (DSTA)
Monterey, CA